

How to deploy on AWS from GitLab

A single application for Lambda, Fargate, EC2, EKS, and ECS



What's inside?

How can you use GitLab for AWS?

- » GitLab + Lambda
- » GitLab + Fargate
- » GitLab + EKS
- » GitLab + ECS
- » GitLab + EC2

Additional AWS functionality in the works

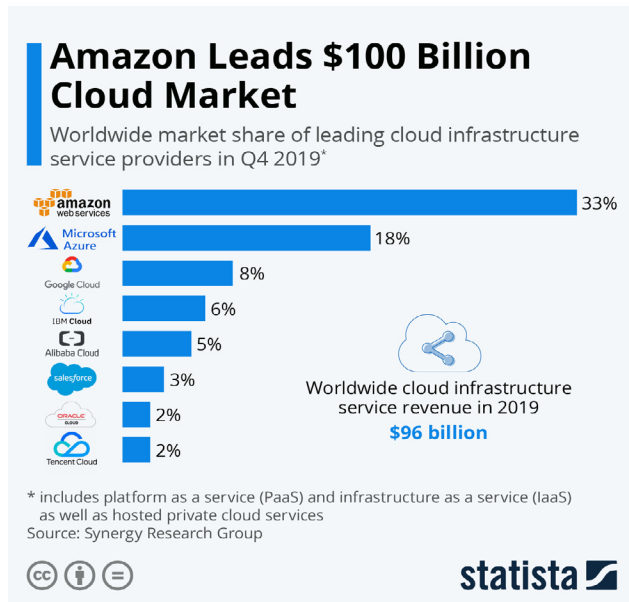
Getting started with GitLab

- » Installing GitLab from AWS Management console
- » Installing GitLab from AWS Marketplace

Benefits of using GitLab for AWS services

Introduction

Amazon Web Services (AWS) is the de facto leader in the worldwide cloud infrastructure market, accounting for nearly 33% in 2019. As of January 2020, AWS offers nearly 160 global cloud-based products including compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security and enterprise applications. Millions of customers, including startups, large-scale enterprises, and government agencies rely on AWS cloud services to power their business.



Developer tools have a high capacity for driving cloud usage and organizations adopt cloud native application development to take advantage of the power of the cloud computing model. Teams utilize cloud native technologies to build and deploy to a limitless number of cloud services, but with small teams managing microservices written in different languages deployed using separate tools, information silos eventually emerge.

Toolchain complexity is a common thread among DevOps teams that are doing cloud

native development today. In a 2019 Forrester survey of IT professionals, they found that many software development teams are struggling with managing and integrating multiple tools. In order to leverage AWS services, development teams can through five or more different tools just to deploy code. Teams then have to manage these tools. Developers end up spending a lot of time maintaining a complex toolchain rather than innovating or building new things.

Many organizations use AWS because it is an all-in-one cloud service, offering everything from storage, to networking, to serverless under one roof. AWS is meant to be comprehensive, and that can bring multiple benefits for its customers:

- » **Simplicity:** All of our cloud data is one place.
- » **Convenience:** AWS has everything we need.
- » **Reliability:** We can rely on AWS to have solutions.
- » **Ease of use:** We don't have to train our team on multiple clouds.

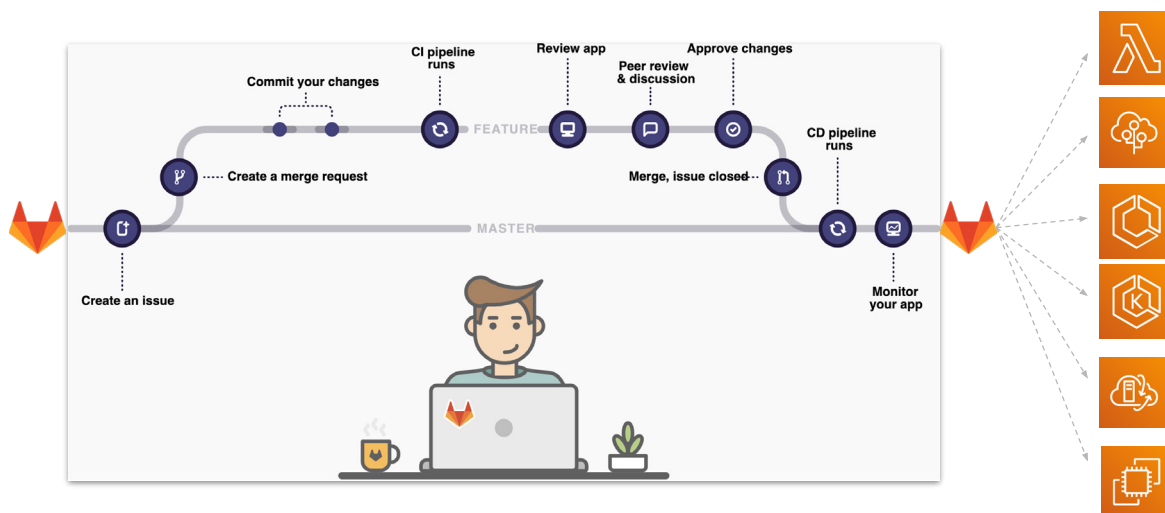
Teams that go “all-in” on AWS do so in order to simplify their cloud needs, but because AWS offers so many cloud services, this simplicity does not come at the cost of functionality.

When it comes to DevOps tools, organizations can also take advantage of this all-in-one meets functionality approach. Instead of bringing in separate tools, integrating them, and maintaining them, DevOps teams can have the entire software development lifecycle in one application with GitLab.

How can you use GitLab for AWS?

What differentiates GitLab from other DevOps applications is that we allow teams to manage the entire software development lifecycle from one interface, eliminating the need for a complicated toolchain. Utilizing [GitLab CI/CD](#), teams can customize deployments to any AWS service. In this white paper, we’ll provide information on how GitLab integrates with five popular AWS services: Lambda, Fargate, EKS, ECS, and EC2.

GitLab: Concurrent Development Workflow



13

GitLab + Lambda

In the Cloud Native Computing Foundation (CNCF) 2019 survey, 41% of respondents use serverless technology. Of those using serverless, 80% use a hosted platform vs. 20% who use installable software. Of those using a hosted platform, the top tool is AWS Lambda (53%).

AWS Lambda is the leader when it comes to building serverless applications. In order to complete a serverless application, organizations will need the following:

- » A computing service
- » A database service
- » An HTTP gateway service

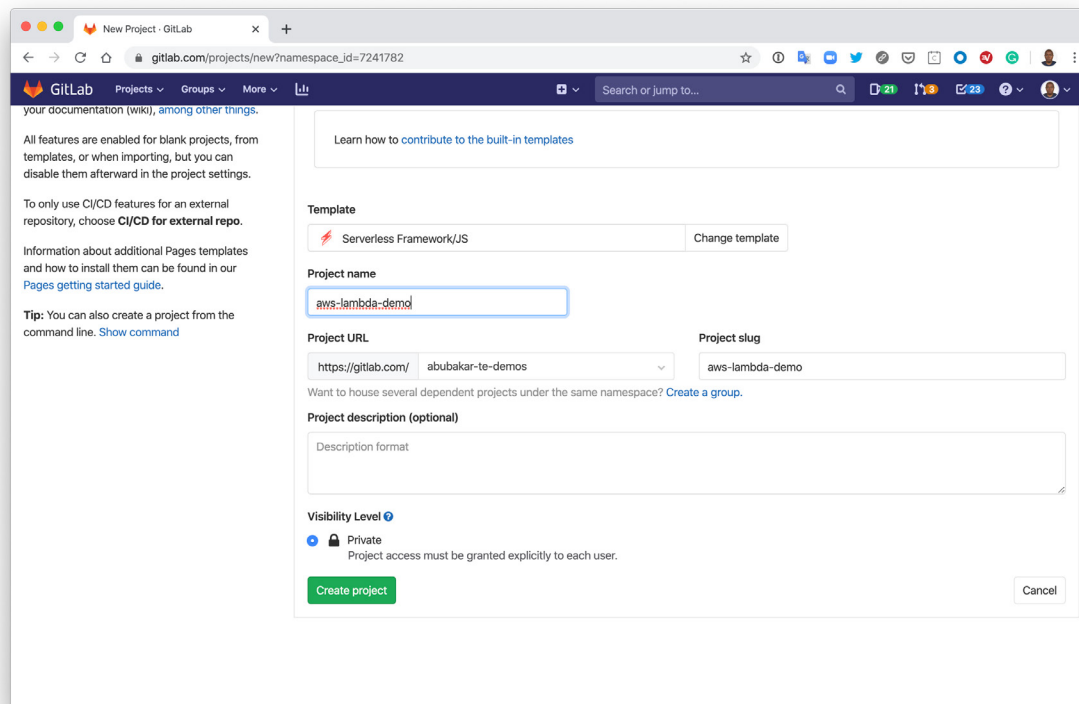
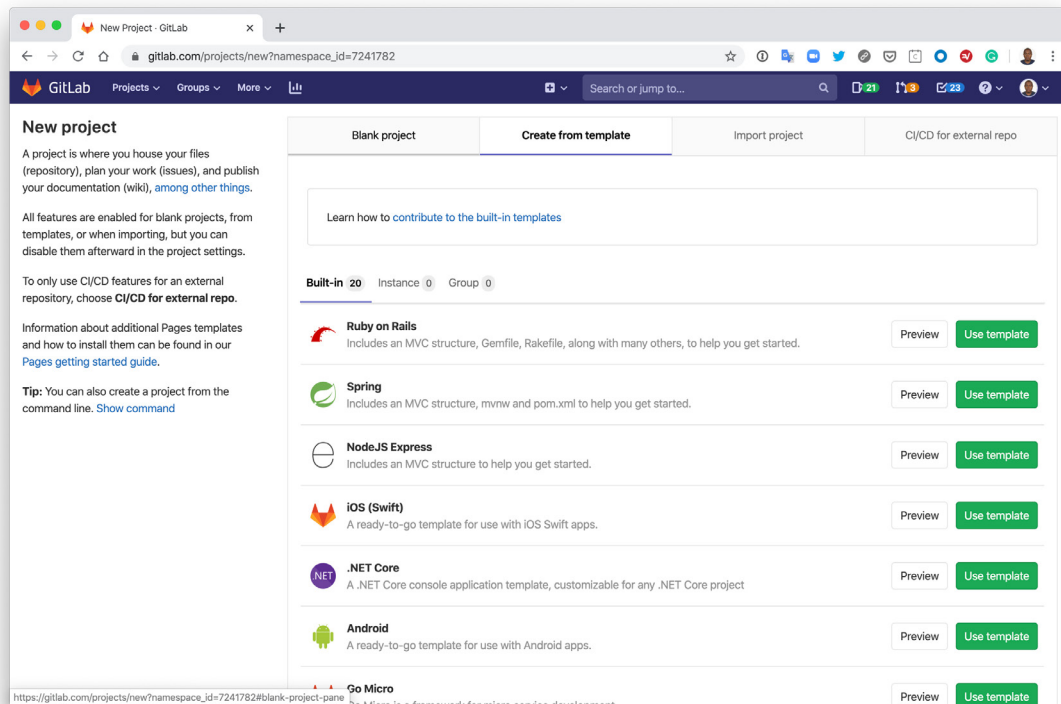
Lambda is an AWS proprietary compute service that integrates with other AWS services as well as open source solutions. The majority of developers building serverless applications today rely on Lambda functions.

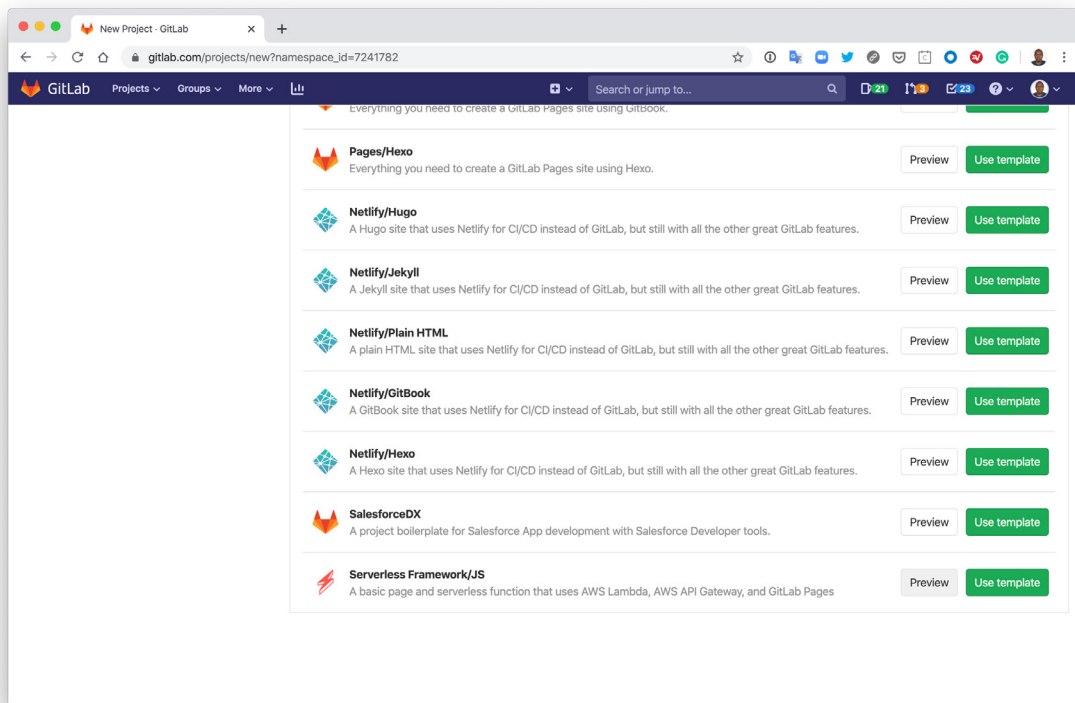
The biggest roadblocks to serverless adoption remain tooling and workflows. Organizations love the scalability and automation of serverless but don't believe that they have the tools to implement it effectively. Enterprises already working within a complicated toolchain are understandably hesitant to add yet another tool into their stack.

GitLab allows users to deploy AWS Lambda functions and create rich serverless applications using a combination of:

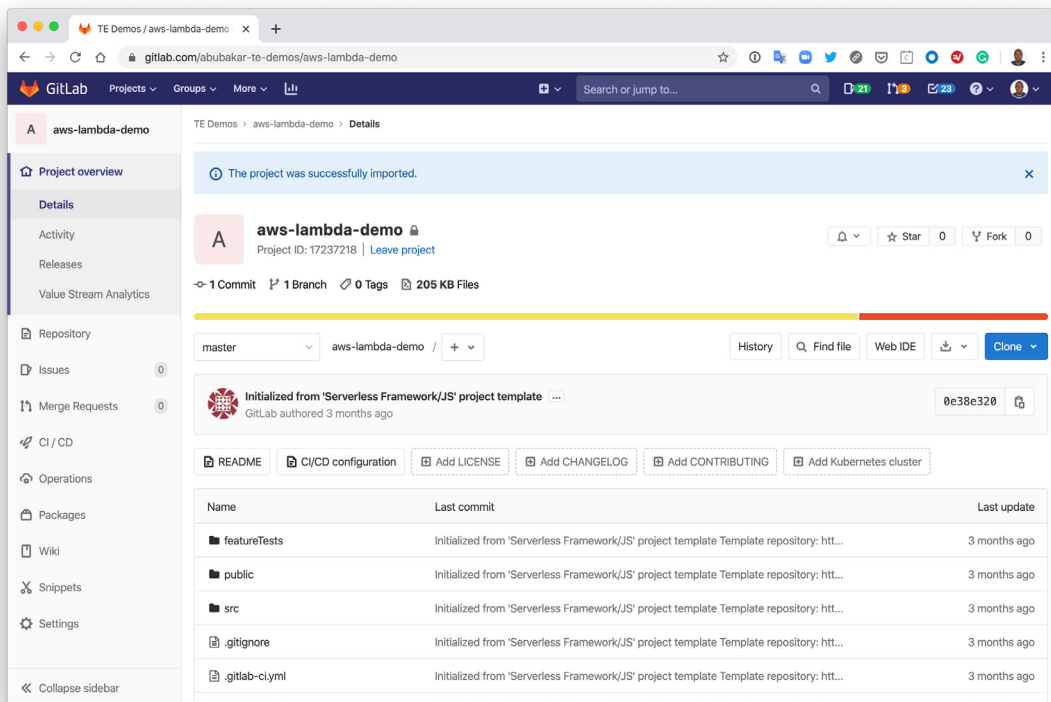
- » Serverless Framework with AWS
- » AWS Serverless Application Model (SAM)
- » GitLab CI/CD

GitLab users have the option to create a project from a Serverless Framework/JS template:





Once the import of the template is complete, you can then work with a sample project:



Next step: create an AWS IAM account to use for authentication:

The screenshot shows the AWS IAM Management Console 'Add user' page. The browser address bar shows 'console.aws.amazon.com/iam/home?region=us-east-2#/users\$new?step=details'. The page has a dark header with the AWS logo and navigation links. The main content area is titled 'Add user' with a progress indicator showing steps 1 through 5. Step 1, 'Set user details', is active. It includes a text input for 'User name*' with the value 'gitlab-lambda-demo' and a link 'Add another user'. Below this is the 'Select AWS access type' section, which asks to 'Select how these users will access AWS'. There are two radio button options: 'Programmatic access' (selected) and 'AWS Management Console access'. The 'Programmatic access' option is described as 'Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.' The 'AWS Management Console access' option is described as 'Enables a password that allows users to sign-in to the AWS Management Console.' At the bottom, there are 'Cancel' and 'Next: Permissions' buttons. The footer contains 'Feedback', 'English (US)', and copyright information.

Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type* ☒ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☐ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

* Required

[Cancel](#) [Next: Permissions](#)

[Feedback](#) [English \(US\)](#) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

Attach the necessary policies to the user:

The screenshot shows the AWS IAM Management Console 'Add user' page, step 2: 'Set permissions'. The browser address bar shows 'console.aws.amazon.com/iam/home?region=us-east-2#/users\$new?step=permissions&accessKey&userName...'. The page has a dark header with the AWS logo and navigation links. The main content area is titled 'Add user' with a progress indicator showing steps 1 through 5. Step 2, 'Set permissions', is active. It includes three buttons: 'Add user to group', 'Copy permissions from existing user', and 'Attach existing policies directly' (highlighted with a blue border). Below these buttons is a 'Create policy' button and a search bar with the text 'lambda'. A table titled 'Showing 13 results' lists various AWS managed policies. The 'AWSLambdaFullAccess' policy is selected with a blue checkmark. At the bottom, there are 'Cancel', 'Previous', and 'Next: Tags' buttons. The footer contains 'Feedback', 'English (US)', and copyright information.

Add user

1 2 3 4 5

Set permissions

[Add user to group](#) [Copy permissions from existing user](#) [Attach existing policies directly](#)

[Create policy](#)

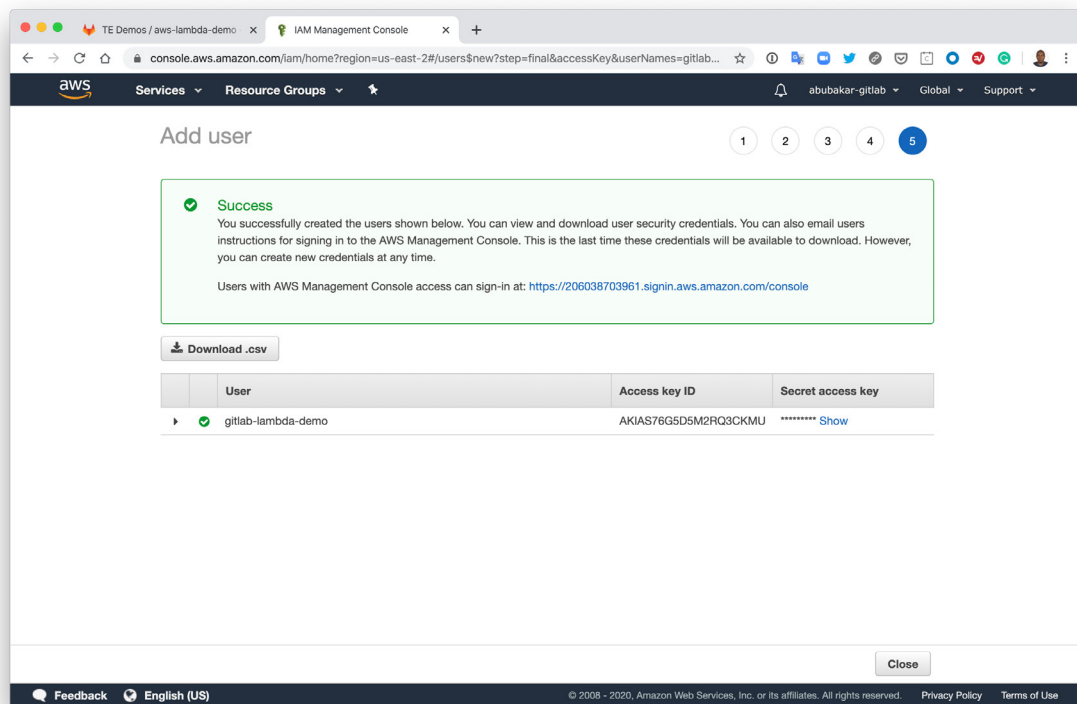
Filter policies Showing 13 results

	Policy name	Type	Used as
<input type="checkbox"/>	AWSCodeDeployRoleForLambda	AWS managed	None
<input type="checkbox"/>	AWSDeepLensLambdaFunctionAccessPolicy	AWS managed	None
<input type="checkbox"/>	AWSLambdaBasicExecutionRole	AWS managed	None
<input type="checkbox"/>	AWSLambdaDynamoDBExecutionRole	AWS managed	None
<input type="checkbox"/>	AWSLambdaENIManagementAccess	AWS managed	None
<input type="checkbox"/>	AWSLambdaExecute	AWS managed	None
<input checked="" type="checkbox"/>	AWSLambdaFullAccess	AWS managed	None
<input type="checkbox"/>	AWSLambdaInvocation-DynamoDB	AWS managed	None

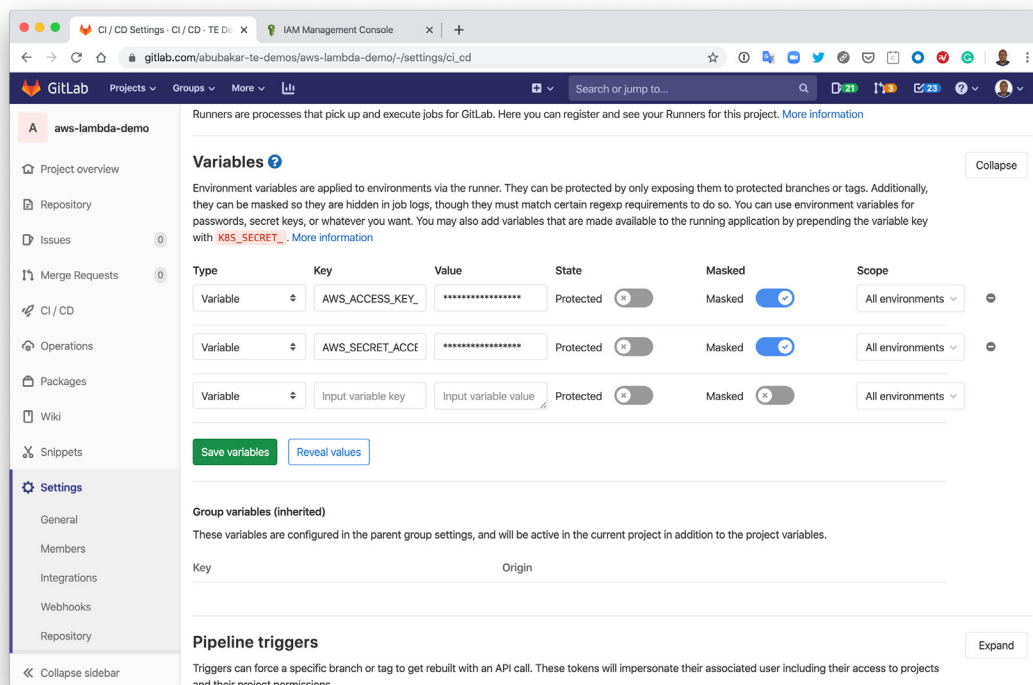
[Cancel](#) [Previous](#) [Next: Tags](#)

[Feedback](#) [English \(US\)](#) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

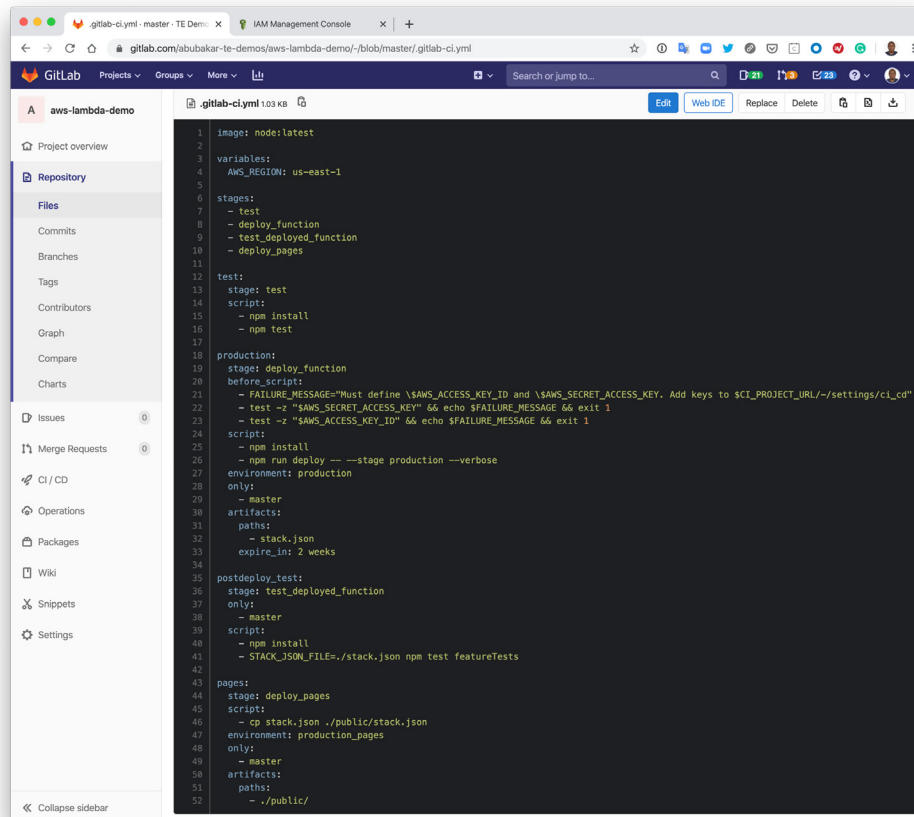
Once these steps are completed, you should have the following summary:



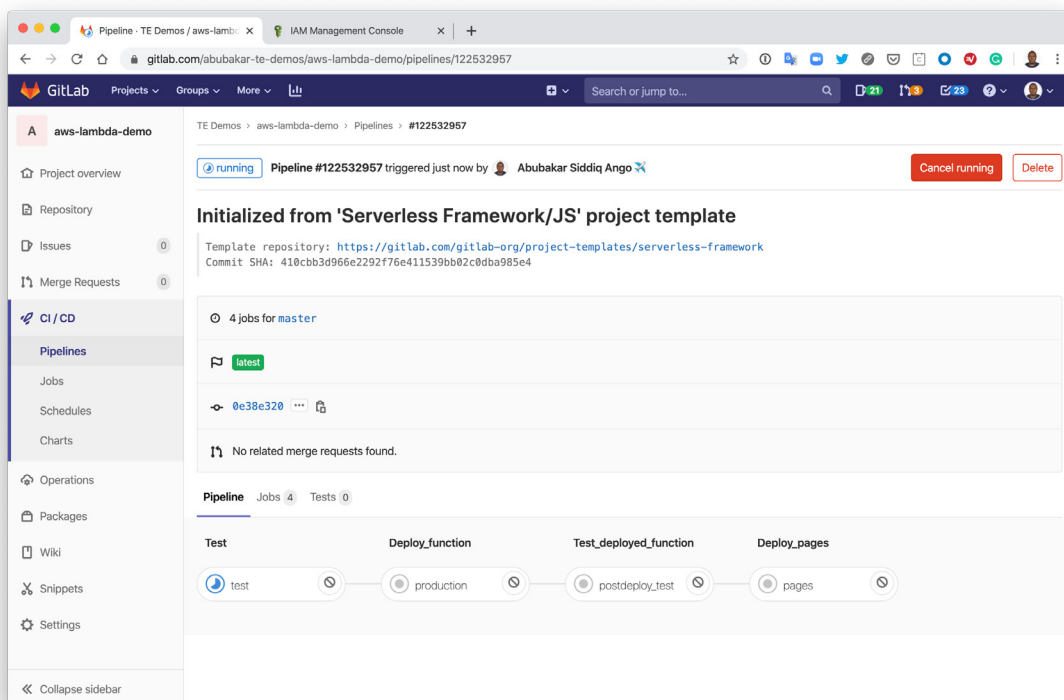
Next step: Add the `AWS_ACCESS_KEY` and `AWS_SECRET_ACCESS_KEY` as CI/CD variables to the project as shown below:



The project sample contains a .gitlab-ci.yml file. Changes made to the repository, or when a pipeline is initiated, four CI jobs (test, production, postdeploy_test and pages) are run as shown below:



```
1 image: node:latest
2
3 variables:
4   AWS_REGION: us-east-1
5
6 stages:
7   - test
8   - deploy_function
9   - test_deployed_function
10  - deploy_pages
11
12 test:
13   stage: test
14   script:
15     - npm install
16     - npm test
17
18 production:
19   stage: deploy_function
20   before_script:
21     - FAILURE_MESSAGE="Must define \${AWS_ACCESS_KEY_ID} and \${AWS_SECRET_ACCESS_KEY}. Add keys to $CI_PROJECT_URL-/settings/ci_cd"
22     - test -z "${AWS_SECRET_ACCESS_KEY}" && echo $FAILURE_MESSAGE && exit 1
23     - test -z "${AWS_ACCESS_KEY_ID}" && echo $FAILURE_MESSAGE && exit 1
24   script:
25     - npm install
26     - npm run deploy -- --stage production --verbose
27   environment: production
28   only:
29     - master
30   artifacts:
31     paths:
32       - stack.json
33     expire_in: 2 weeks
34
35 postdeploy_test:
36   stage: test_deployed_function
37   only:
38     - master
39   script:
40     - npm install
41     - STACK_JSON_FILE=./stack.json npm test featureTests
42
43 pages:
44   stage: deploy_pages
45   script:
46     - cp stack.json ./public/stack.json
47   environment: production_pages
48   only:
49     - master
50   artifacts:
51     paths:
52       - ./public/
```



GitLab Pipeline view for 'aws-lambda-demo' project. The pipeline is currently running (Pipeline #122532957). It was triggered by Abubakar Siddiq Ango. The pipeline is initialized from the 'Serverless Framework/JS' project template.

Template repository: <https://gitlab.com/gitlab-org/project-templates/serverless-framework>
Commit SHA: 410cbb3d966e2292f76e411539bb02c0dba985e4

4 jobs for master

latest

0e38e320 ...

No related merge requests found.

Pipeline Jobs 4 Tests 0

Test Deploy_function Test_deployed_function Deploy_pages

test production postdeploy_test pages

GitLab Projects Groups More

aws-lambda-demo

Project overview
Repository
Issues
Merge Requests

CI / CD
Pipelines
Jobs
Schedules

Operations
Packages
Analytics
Wiki
Snippets
Settings

TE Demos > aws-lambda-demo > Pipelines > #122532957

passed Pipeline #122532957 triggered 22 hours ago by Abubakar Siddiq Ango

Initialized from 'Serverless Framework/JS' project template

Template repository: <https://gitlab.com/gitlab-org/project-templates/serverless-framework>
Commit SHA: 410cbb3d966e2292f76e411539bb02c0dba905e4

9 jobs for **master** in 7 minutes and 4 seconds (queued for 162 minutes and 13 seconds)

latest

0e38e320

No related merge requests found.

Pipeline Jobs 9 Tests 0

Test Deploy_function Test_deployed_function Deploy_pages Deploy

test production postdeploy_test pages pages:deploy

Once the production job completes, the URL of the Lambda endpoint is created:

GitLab Projects Groups More

aws-lambda-demo

Project overview
Repository
Issues
Merge Requests

CI / CD
Pipelines
Jobs
Schedules

Operations
Packages
Analytics
Wiki
Snippets
Settings

Search or jump to...

```

82 CloudFormation - CREATE_IN_PROGRESS - AWS::ApiGateway::Deployment - ApiGatewayDeployment1583170
83 CloudFormation - CREATE_COMPLETE - AWS::ApiGateway::Deployment - ApiGatewayDeployment1583170045
84 CloudFormation - CREATE_COMPLETE - AWS::Lambda::Permission - HelloLambdaPermissionApiGateway
85 CloudFormation - UPDATE_COMPLETE_CLEANUP_IN_PROGRESS - AWS::CloudFormation::Stack - gitlab-exam
86 CloudFormation - UPDATE_COMPLETE - AWS::CloudFormation::Stack - gitlab-example-production
87 Serverless: Stack update finished...
88 Service Information
89 service: gitlab-example
90 stage: production
91 region: us-east-1
92 stack: gitlab-example-production
93 resources: 12
94 api keys:
95 None
96 endpoints:
97 GET - https://8sxglpasw9.execute-api.us-east-1.amazonaws.com/production/hello
98 functions:
99 hello: gitlab-example-production-hello
100 layers:
101 None
102 Stack Outputs
103 HelloLambdaFunctionQualifiedArn: arn:aws:lambda:us-east-1:206038703961:function:gitlab-example-
104 ServiceEndpoint: https://8sxglpasw9.execute-api.us-east-1.amazonaws.com/production
105 ServerlessDeploymentBucketName: gitlab-example-productio-serverlessdeploymentbuck-1flxdlsfj5ja
106 Serverless: Run the "serverless" command to setup monitoring, troubleshooting and testing.
107 Serverless: Stack Output processed with handler: src/handler.hello
108 Serverless: Stack Output saved to file: stack.json
109
110 Uploading artifacts...
111 stack.json: found 1 matching files
112 Uploading artifacts to coordinator... ok id=456475529 responseStatus=201 Created tok
113 en=689riPPR
114 Job succeeded

```

00:03

Collapse sidebar

In this example, when a GET request is sent to the URL, it returns the following sample text:

The screenshot displays a REST client interface with the following components:

- Environment:** No Environment
- Request Method:** GET
- URL:** `https://8sxlpsw9.execute-api.us-east-1.amazonaws.com/production/hello`
- Buttons:** Send
- Tabs:** Params, Authorization, Headers (7), Body, Pre-request Script, Tests, Settings
- Query Params Table:**

KEY	VALUE	DESCRIPTION
Key	Value	Description
- Status Bar:** Status: 200 OK, Time: 1005ms, Size: 580 B
- Response Body:**
 - Tabs:** Pretty, Raw, Preview, Visualize
 - Format:** JSON
 - Content:**

```
1 {
2   "message": "AWS Lambda using GitLab",
3   "params": null
4 }
```


GitLab + SAM

AWS SAM, at the highest level, is an open source framework for building serverless applications on AWS. It can be considered an extension to CloudFormation that makes it easier to define and deploy AWS resources – such as Lambda functions, API Gateway APIs and DynamoDB tables – commonly used in serverless applications.

In addition to its templating capabilities, SAM also includes a CLI for testing and deployment, though some of the CLI commands are just aliases to underlying CloudFormation calls. In this sample project, we used the AWS CloudFormation CLI to build and deploy our SAM application.

We've included a sample SAM application called "AWS News" in the `{{cookiecutter.project_name}}` folder, but you can use the `.gitlab-ci.yml` file at the top of this repo with any SAM application.

First, install AWS SAM CLI:

```
abubakar_gitlab@Abubakars-MacBook-Pro: ~  
/usr/local/Cellar/sqlite/3.31.1: 11 files, 4MB  
=> Installing aws/tap/aws-sam-cli dependency: python  
=> Downloading https://homebrew.bintray.com/bottles/python-3.7.6_1.catalina.bottle.tar.gz  
=> Downloading from https://akamai.bintray.com/38/3871ef8b53270576c46489ae397f245b84772c4050  
##### 100.0%  
=> Pouring python-3.7.6_1.catalina.bottle.tar.gz  
=> /usr/local/Cellar/python/3.7.6_1/bin/python3 -s setup.py --no-user-cfg install --force --  
=> /usr/local/Cellar/python/3.7.6_1/bin/python3 -s setup.py --no-user-cfg install --force --  
=> /usr/local/Cellar/python/3.7.6_1/bin/python3 -s setup.py --no-user-cfg install --force --  
=> Caveats  
Python has been installed as  
  /usr/local/bin/python3  
  
Unversioned symlinks `python`, `python-config`, `pip` etc. pointing to  
`python3`, `python3-config`, `pip3` etc., respectively, have been installed into  
  /usr/local/opt/python/libexec/bin  
  
You can install Python packages with  
  pip3 install <package>  
They will install into the site-package directory  
  /usr/local/lib/python3.7/site-packages  
  
See: https://docs.brew.sh/Homebrew-and-Python  
=> Summary  
/usr/local/Cellar/python/3.7.6_1: 3,977 files, 61MB  
=> Installing aws/tap/aws-sam-cli  
=> Downloading https://github.com/aws-labs/aws-sam-cli/releases/download/v0.43.0/aws-sam-cli  
=> Downloading from https://github-production-release-asset-2e65be.s3.amazonaws.com/92205085  
##### 100.0%  
=> Pouring aws-sam-cli-0.43.0.sierra.bottle.tar.gz  
/usr/local/Cellar/aws-sam-cli/0.43.0: 3,975 files, 63.2MB  
=> `brew cleanup` has not been run in 30 days, running now...  
Removing: /Users/abubakar_gitlab/Library/Caches/Homebrew/dart--2.7.0.zip... (205.7MB)  
Removing: /Users/abubakar_gitlab/Library/Caches/Homebrew/gmp--6.1.2_2.catalina.bottle.1.tar.gz... (996  
.4KB)  
Removing: /Users/abubakar_gitlab/Library/Caches/Homebrew/helm--3.0.2.catalina.bottle.tar.gz... (12.1MB)
```

Run the SAM init command to generate AWS SAM project using a hello-world example:

```
abubakar_gitlab@Abubakars-MacBook-Pro: ~/work/gitlab/demo
> sam init -r python3.8 -n gitlab-aws-same-demo --app-template "hello-world"

Cloning app templates from https://github.com/aws-labs/aws-sam-cli-app-templates.git

-----
Generating application:
-----
Name: gitlab-aws-same-demo
Runtime: python3.8
Dependency Manager: pip
Application Template: hello-world
Output Directory: .

Next steps can be found in the README file at ./gitlab-aws-same-demo/README.md
```

A folder is created for the project, in our case named `gitlab-aws-sam-demo`. Next, we create a `.gitlab-ci.yml` file to build and deploy the project.

[illegible]

Next, we run git init and push to GitLab:

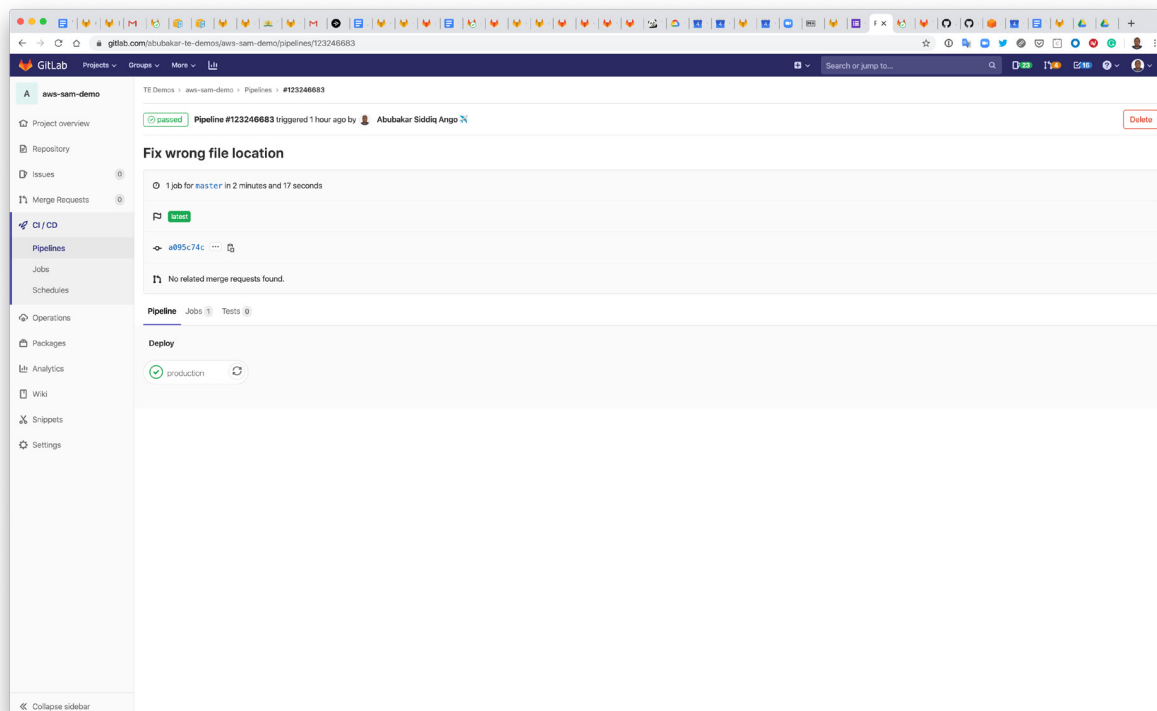
```
abubakar_gitlab@Abubakars-MacBook-Pro: ~/work/gitlab/demo/gitlab-aws-sam-demo
Application Template: hello-world
Output Directory: .

Next steps can be found in the README file at ./gitlab-aws-same-demo/README.md

> ls
aws-sam-demo      gitlab-aws-same-demo
> mv gitlab-aws-same-demo gitlab-aws-sam-demo
> cd gitlab-aws-sam-demo
> ls -al
total 32
drwxr-xr-x  8 abubakar_gitlab  staff   256 Mar  4 16:34 .
drwxr-xr-x  4 abubakar_gitlab  staff   128 Mar  4 16:34 ..
-rw-r--r--  1 abubakar_gitlab  staff  3730 Mar  4 16:34 .gitignore
-rw-r--r--  1 abubakar_gitlab  staff  7620 Mar  4 16:34 README.md
drwxr-xr-x  3 abubakar_gitlab  staff    96 Mar  4 16:34 events
drwxr-xr-x  5 abubakar_gitlab  staff   160 Mar  4 16:34 hello_world
-rw-r--r--  1 abubakar_gitlab  staff  1649 Mar  4 16:34 template.yaml
drwxr-xr-x  3 abubakar_gitlab  staff    96 Mar  4 16:34 tests
> cp ../aws-sam-demo/.gitlab-ci.yml .
> vi .gitlab-ci.yml
> ls -al
total 40
drwxr-xr-x  9 abubakar_gitlab  staff   288 Mar  4 16:36 .
drwxr-xr-x  4 abubakar_gitlab  staff   128 Mar  4 16:34 ..
-rw-r--r--  1 abubakar_gitlab  staff  3730 Mar  4 16:34 .gitignore
-rw-r--r--  1 abubakar_gitlab  staff   470 Mar  4 16:35 .gitlab-ci.yml
-rw-r--r--  1 abubakar_gitlab  staff  7620 Mar  4 16:34 README.md
drwxr-xr-x  3 abubakar_gitlab  staff    96 Mar  4 16:34 events
drwxr-xr-x  5 abubakar_gitlab  staff   160 Mar  4 16:34 hello_world
-rw-r--r--  1 abubakar_gitlab  staff  1649 Mar  4 16:34 template.yaml
drwxr-xr-x  3 abubakar_gitlab  staff    96 Mar  4 16:34 tests
> git init
git remote add origin git@gitlab.com:abubakar-te-demos/gitlab-aws-sam-demo.git
git add .
Initialized empty Git repository in /Users/abubakar_gitlab/work/gitlab/demo/gitlab-aws-sam-demo/.git/

~/.work/gitlab/demo/gitlab-aws-sam-demo on master +10 at 16:37:46
```

Once the pipeline is complete, it outputs the URL of the serverless endpoint as shown below:



```

175 Region : us-east-2
176 Confirm changeset : False
177 Deployment s3 Bucket : abubakar-aws-sam-demo
178 Capabilities : ["CAPABILITY_IAM"]
179 Parameter overrides : {}
180 Initiating deployment
181
182 Uploading to 4ca9612b6a7f5c25fca91330a0b76df.template 1069 / 1069.0 (100.00%)
183 Waiting for changeset to be created..
184 CloudFormation stack changeset
185
186 Operation LogicalResourceId ResourceType
187
188 + Modify HelloWorldFunction AWS::Lambda::Function
189 + Modify ServerlessRestApi AWS::ApiGateway::RestApi
190
191 Changeset created successfully. arn:aws:cloudformation:us-east-2:206838783961:changeset/samcli-deploy1583336399/4573a982-1ba5-4c88-b1e6-6e351bf3be2
192 2020-03-04 15:40:04 - Waiting for stack create/update to complete
193 CloudFormation events from changeset
194
195 ResourceStatus ResourceType LogicalResourceId ResourceStatusReason
196
197 UPDATE_IN_PROGRESS AWS::Lambda::Function HelloWorldFunction -
198 UPDATE_COMPLETE AWS::Lambda::Function HelloWorldFunction -
199 UPDATE_COMPLETE_CLEANUP AWS::CloudFormation::Stack gitlabpoc -
200 P_IN_PROGRESS Stack gitlabpoc -
201 UPDATE_COMPLETE AWS::CloudFormation::Stack gitlabpoc -
202
203 CloudFormation outputs from deployed stack
204
205 Outputs
206
207
208 Key HelloWorldFunctionIamRole
209 Description Implicit IAM Role created for Hello World function
210 Value arn:aws:iam::206838783961:role/gitlabpoc-
211 HelloWorldFunctionRole-3XG5J10GCP96
212 Key HelloWorldApi
213 Description API Gateway endpoint URL for Prod stage for Hello World Function
214 Value https://xlzlb1s7.execute-api.us-east-2.amazonaws.com/Prod/hello/
215 Key HelloWorldFunction
216 Description Hello World Lambda Function ARN
217 Value arn:aws:lambda:us-east-2:206838783961:function:gitlabpoc-
218 HelloWorldFunction-1PQ2KJ2Z2Y5C7
219
220 Successfully created/updated stack - gitlabpoc in us-east-2
221 Job succeeded

```

Using curl with the Service endpoint shows the output required, which is “Hello World”.

```

abubakar_gitlab@Abubakars-MacBook-Pro: ~/work/gitlab/demo/gitlab-aws-sam-demo
-rw-r--r-- 1 abubakar_gitlab staff 7620 Mar 4 16:34 README.md
drwxr-xr-x 3 abubakar_gitlab staff 96 Mar 4 16:34 events
drwxr-xr-x 5 abubakar_gitlab staff 160 Mar 4 16:34 hello_world
-rw-r--r-- 1 abubakar_gitlab staff 1649 Mar 4 16:34 template.yaml
drwxr-xr-x 3 abubakar_gitlab staff 96 Mar 4 16:34 tests
> git init
git remote add origin git@gitlab.com:abubakar-te-demos/gitlab-aws-sam-demo.git
git add .
Initialized empty Git repository in /Users/abubakar_gitlab/work/gitlab/demo/gitlab-aws-sam-demo/.git/
> git commit -m "Initial Commit"
[master (root-commit) b7e47e7] Initial Commit
10 files changed, 606 insertions(+)
create mode 100644 .gitignore
create mode 100644 .gitlab-ci.yml
create mode 100644 README.md
create mode 100644 events/event.json
create mode 100644 hello_world/__init__.py
create mode 100644 hello_world/app.py
create mode 100644 hello_world/requirements.txt
create mode 100644 template.yaml
create mode 100644 tests/unit/__init__.py
create mode 100644 tests/unit/test_handler.py
> git push -u origin master
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (15/15), 8.16 KiB | 2.72 MiB/s, done.
Total 15 (delta 1), reused 0 (delta 0)
To gitlab.com:abubakar-te-demos/gitlab-aws-sam-demo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
> ls
README.md  events  hello_world  template.yaml  tests
> curl https://xlzlb1s7.execute-api.us-east-2.amazonaws.com/Prod/hello/
{"message": "hello world"}

```

Additional resources:

- » [serverless.yaml Configuration](#)
- » [Document how to deploy serverless functions to AWS Lambda](#)
- » [Deploying AWS Lambda function using GitLab CI/CD](#)
- » [AWS Serverless Application Model \(SAM\)](#)
- » [Serverless Framework](#)
- » [TriggerMesh KLR as a CI/CD provider](#)

GitLab + Fargate

According to a [recent Datadog market analysis](#), among companies running containers in AWS, 19% use [AWS Fargate](#) – up from 14% the year before. Fargate’s rapid growth has helped Amazon Elastic Container Service keep pace with the continuing adoption of Kubernetes in AWS environments.

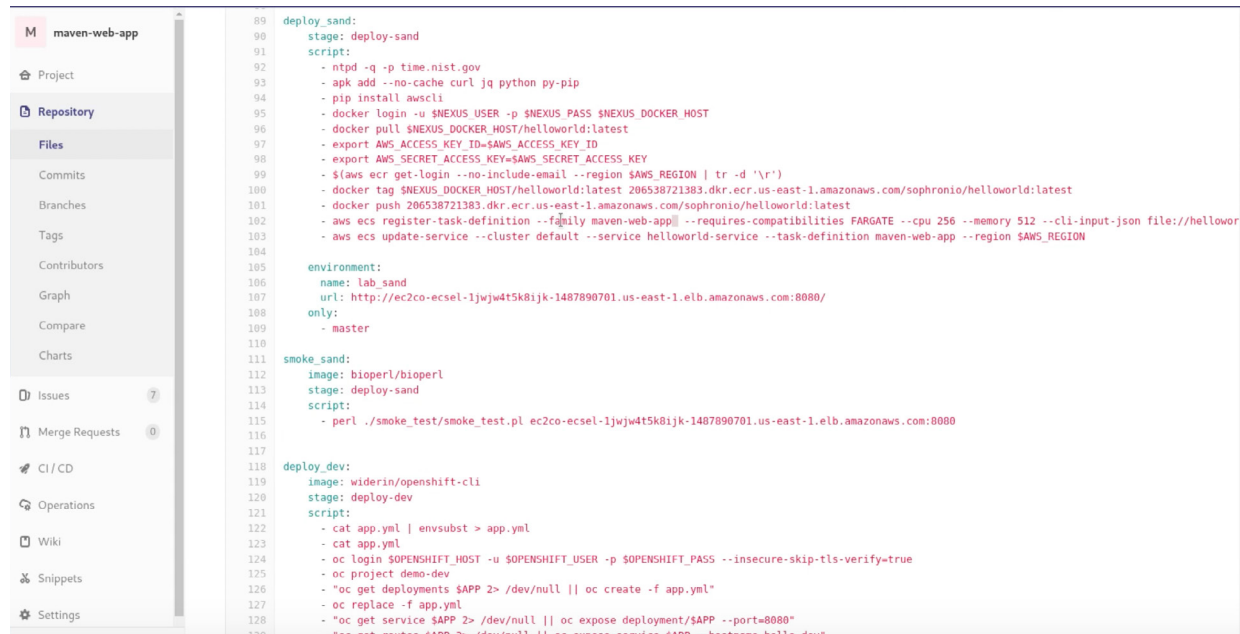
AWS Fargate is a serverless compute engine for containers that works with both [Amazon Elastic Container Service \(ECS\)](#) and [Amazon Elastic Kubernetes Service \(EKS\)](#). Fargate primarily removes the need to provision and manage servers and allows developers to specify resources for each application.

To get started using GitLab and Fargate you’ll need your AWS stack in place:

1. An Amazon ECS [cluster](#)
2. A service that belongs to the cluster that runs Fargate tasks
3. An accessible [container registry](#) for Docker images
4. An Amazon ECS [task definition](#) that references a Docker image stored in your registry that defines CPU and memory requirements
5. An [application load balancer](#) that redirects requests to healthy targets in a target group
6. A [target group](#)

Once you have these pieces in place, it’s just a matter of creating a GitLab CI/CD pipeline and updating the `.gitlab-ci.yml` file.

Here is a sample script for a dev environment that specifies task definitions for a Fargate cluster:



The screenshot shows a GitLab repository interface for a project named 'maven-web-app'. The left sidebar contains navigation links: Project, Repository, Files, Commits, Branches, Tags, Contributors, Graph, Compare, Charts, Issues (7), Merge Requests (0), CI/CD, Operations, Wiki, Snippets, and Settings. The main content area displays a script for a Fargate cluster, with line numbers 89 to 130 visible. The script defines three stages: 'deploy_sand', 'smoke_sand', and 'deploy_dev'. The 'deploy_sand' stage sets up a Docker environment with AWS credentials and pushes a Docker image. The 'smoke_sand' stage runs a smoke test. The 'deploy_dev' stage uses OpenShift CLI to deploy the application to a Fargate cluster.

```
89 deploy_sand:
90   stage: deploy-sand
91   script:
92     - ntpd -q -p time.nist.gov
93     - apk add --no-cache curl jq python py-pip
94     - pip install awscli
95     - docker login -u $NEXUS_USER -p $NEXUS_PASS $NEXUS_DOCKER_HOST
96     - docker pull $NEXUS_DOCKER_HOST/helloworld:latest
97     - export AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID
98     - export AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY
99     - $(aws ecr get-login --no-include-email --region $AWS_REGION | tr -d '\r')
100     - docker tag $NEXUS_DOCKER_HOST/helloworld:latest 206538721383.dkr.ecr.us-east-1.amazonaws.com/sophonio/helloworld:latest
101     - docker push 206538721383.dkr.ecr.us-east-1.amazonaws.com/sophonio/helloworld:latest
102     - aws ecs register-task-definition --family maven-web-app --requires-compatibilities FARGATE --cpu 256 --memory 512 --cli-input-json file://hellowor
103     - aws ecs update-service --cluster default --service helloworld-service --task-definition maven-web-app --region $AWS_REGION
104
105   environment:
106     name: lab_sand
107     url: http://ec2co-ecsel-1jwjw4t5k8ijk-1487890701.us-east-1.elb.amazonaws.com:8080/
108   only:
109     - master
110
111   smoke_sand:
112     image: bioperl/bioperl
113     stage: deploy-sand
114     script:
115       - perl ./smoke_test/smoke_test.pl ec2co-ecsel-1jwjw4t5k8ijk-1487890701.us-east-1.elb.amazonaws.com:8080
116
117   deploy_dev:
118     image: widerin/openshift-cli
119     stage: deploy-dev
120     script:
121       - cat app.yml | envsubst > app.yml
122       - cat app.yml
123       - oc login $OPENSHIFT_HOST -u $OPENSHIFT_USER -p $OPENSHIFT_PASS --insecure-skip-tls-verify=true
124       - oc project demo-dev
125       - "oc get deployments $APP 2> /dev/null || oc create -f app.yml"
126       - oc replace -f app.yml
127       - "oc get service $APP 2> /dev/null || oc expose deployment/$APP --port=8080"
128       - "oc get routes $APP 2> /dev/null || oc expose service $APP --hostname=hello.dev"
```

Additional resources:

See how the team at Web Captioner was able to use GitLab and AWS for one-click Fargate deployments.

[Read their story](#)

GitLab + EKS

Amazon Elastic Kubernetes Service (EKS) is a fully managed Kubernetes service. The introduction of Amazon Elastic Kubernetes Service (EKS) was widely applauded as it streamlines the abstraction of the complexities in an environment and makes creating and managing Kubernetes clusters easier with more granular controls around security and straightforward policies of how resources are used.

GitLab strives to increase developer productivity by automating repetitive tasks and allowing developers to focus on business logic. We recently introduced support for auto-creating Kubernetes clusters on Amazon EKS. GitLab also gives you the power to achieve the following use cases and more:

- » Highly scalable CI/CD system using GitLab Runner: There are times like holidays when little to no changes to code are pushed to production, so why keep resources tied down? With the Amazon EKS integration with GitLab, you can install GitLab Runner with just a click and your CI/CD will run effortlessly without worrying about running out of resources.
- » Shared Cluster: Maintaining multiple Kubernetes clusters can be a pain and capital intensive. With Amazon EKS, GitLab allows you to setup a cluster at Instance, Group and Project levels. Kubernetes Namespaces are created for each GitLab project when the Amazon EKS is integrated at Instance and Project level, allowing isolation and ensuring security.
- » Review Apps: Reviewing changes to code or design can be tricky, you'll need to check out your branch and run the code in a test environment. GitLab integrated with Amazon EKS deploys your app with new changes to a dynamic environment and all you need to do is click on a "View App" button to review changes.
- » AutoDevOps detects, builds, tests, deploys, and monitors your applications, leveraging the Amazon EKS integration. In this section, we will deploy a sample application to the Amazon EKS cluster we create using AutoDevOps.

One-time setup on AWS to access resources

First, we need to create a "provision" role and a "service" role on AWS to grant GitLab access to AWS resources and set up the necessary permissions to create and manage EKS clusters. You only need to perform these steps once and you can reuse them anytime you want to perform another integration or create more clusters.

Step 1 - Create provision role

To grant GitLab access to your AWS resources, a “provision role” is required.

1. Access GitLab Kubernetes Integration Page by clicking on the “Kubernetes” menu for groups and Operations > Kubernetes menu for projects and click the “Add Kubernetes Cluster” button.
2. Select “Amazon EKS” in the options provided under the “Create new cluster on EKS” tab.
3. You are provided with an Account and External ID to use for authentication. Make note of these values to be used in a later step.

TE Demos > Kubernetes

Add a Kubernetes cluster integration


Adding a Kubernetes cluster to your group will automatically share the cluster across all your projects. Use review apps, deploy your applications, and easily run your pipelines for all projects using the same cluster.


[Learn more about group Kubernetes clusters](#)

Create new cluster on EKS

Add existing cluster

Create cluster on


Amazon EKS


Google GKE

Authenticate with Amazon Web Services

You must grant access to your organization's AWS resources in order to create a new EKS cluster. To grant access, create a provision role using the account and external ID below and provide us the ARN.

Account ID

855262394183

External ID

94a9f5703b0c7d7b31b16c8a28843c78b38b6102

Create a provision role on [Amazon Web Services](#) using the account and external ID above. [More information](#)

Provision Role ARN

The Amazon Resource Name (ARN) associated with your role. If you do not have a provision role, first create one on [Amazon Web Services](#) using the above account and external IDs. [More information](#)

Authenticate with AWS

4. Open IAM Management Console in another tab and click on “Create Role”

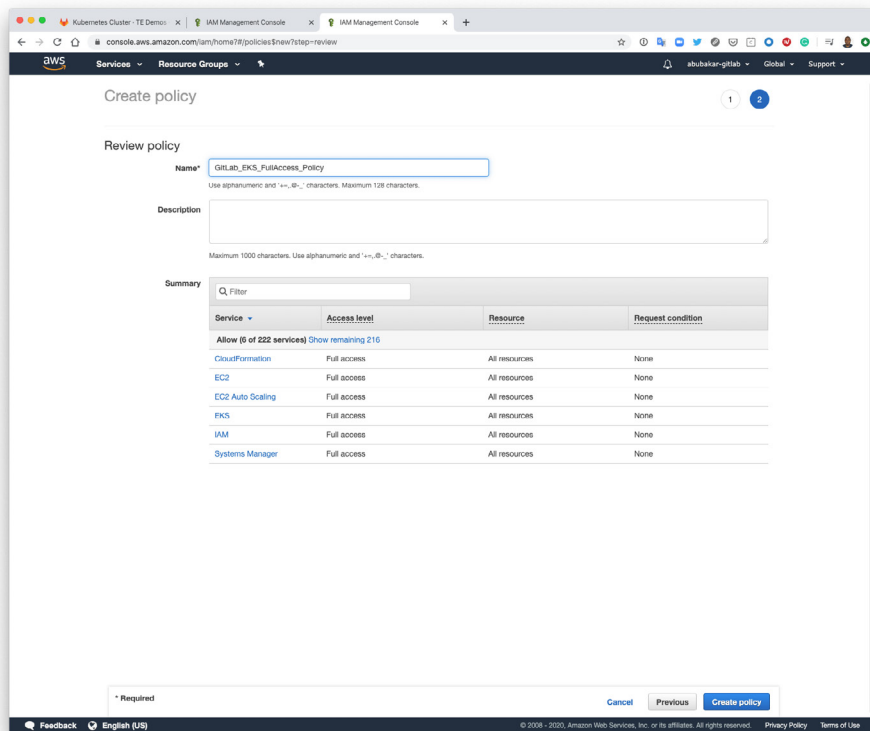
- Click on the “Another AWS account” tab and provide the Account and External ID obtained from GitLab and click Next to set permissions as shown below:

The screenshot shows the AWS IAM console 'Create role' page. The 'Another AWS account' tab is selected. The 'Account ID' field contains '855262394183'. The 'Require external ID' checkbox is checked, and the 'External ID' field contains '1d1fcd8a28843c78b38b6102'. The 'Next: Permissions' button is visible at the bottom right.

On the permissions page, click on “Create policy.”
This will open a new tab where you can set either of the permissions below using JSON:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:*",
        "cloudformation:*",
        "ec2:*",
        "eks:*",
        "iam:*",
        "ssm:*"
      ],
      "Resource": "*"
    }
  ]
}
```

This gives GitLab full access to create and manage resources, as seen in the image below:



If you prefer limited permission, you can give GitLab the ability to create resources, but not delete them with the JSON snippet below. The drawback here is if an error is encountered during the creation process, changes will not be rolled back and you must remove resources manually. You can do this by deleting the relevant CloudFormation stack.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeScalingActivities",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:CreateLaunchConfiguration",
        "autoscaling:DescribeLaunchConfigurations",
        "cloudformation:CreateStack",
        "cloudformation:DescribeStacks",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:CreateSecurityGroup",
        "ec2:createTags",
        "ec2:DescribeImages",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeRegions",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "eks:CreateCluster",
        "eks:DescribeCluster",
        "iam:AddRoleToInstanceProfile",
        "iam:AttachRolePolicy",
        "iam:CreateRole",
        "iam:CreateInstanceProfile",
        "iam:CreateServiceLinkedRole",
        "iam:GetRole",
        "iam:ListRoles",
        "iam:PassRole",
        "ssm:GetParameters"
      ],
      "Resource": "*"
    }
  ]
}

```

6. The image below visualizes what permissions are granted:

Summary

Q Filter			
Service ▼	Access level	Resource	Request condition
Allow (6 of 222 services) Show remaining 216			
CloudFormation	Limited: List, Write	All resources	None
EC2	Limited: List, Write, Tagging	All resources	None
EC2 Auto Scaling	Limited: List, Write, Tagging	All resources	None
EKS	Limited: Read, Write	All resources	None
IAM	Limited: List, Read, Write, Permissions management	All resources	None
Systems Manager	Limited: Read	All resources	None

7. Once the policy is created, return to the “Create Role” browser tab and refresh to see the policy we created listed. Select the policy and click “Next.”
8. In the Tags section, we don’t need to set any Tags, except if it’s required in your organization. Let’s proceed to Review.
9. Specify a Name for your new Role. You will see the policy we created listed under policies and click “Create Role” to complete the process.
10. Click on the new Role you created in the list of Roles to view its details. You may have to search for it in the list of Roles if it’s not listed in the first view. Copy the Role ARN provided – we will need it on the GitLab Kubernetes Integration page.

Step 2 - Create service role


The Service Role is required to allow Amazon EKS and the Kubernetes control plane to manage AWS resources on your behalf.


1. In the IAM Management Console, click on “Create Role” and select the “AWS service” tab.
2. Select EKS in the list of services and Use Cases as shown below and click Next.

Create role


1 2 3 4

Select type of trusted entity

**AWS service**
EC2, Lambda and others

**Another AWS account**
Belonging to you or 3rd party

**Web identity**
Cognito or any OpenID provider

**SAML 2.0 federation**
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose a use case

Common use cases

EC2

Allows EC2 instances to call AWS services on your behalf.

Lambda

Allows Lambda functions to call AWS services on your behalf.

Or select a service to view its use cases

API Gateway	CodeDeploy	EMR	KMS	RoboMaker
AWS Backup	CodeGuru	ElastiCache	Kinesis	S3
AWS Chatbot	CodeStar Notifications	Elastic Beanstalk	Lambda	SMS
AWS Support	Comprehend	Elastic Container Service	Lex	SNS
Amplify	Config	Elastic Transcoder	License Manager	SWF
AppStream 2.0	Connect	ElasticLoadBalancing	Machine Learning	SageMaker
AppSync	DMS	Forecast	Macie	Security Hub
Application Auto Scaling	Data Lifecycle Manager	Global Accelerator	MediaConvert	Service Catalog
Application Discovery Service	Data Pipeline	Glue	Migration Hub	Step Functions
Batch	DataSync	Greengrass	OpsWorks	Storage Gateway
Chime	DeepLens	GuardDuty	Personalize	Textract
CloudFormation	Directory Service	Health Organizational View	QLDB	Transfer
CloudHSM	DynamoDB	IAM Access Analyzer	RAM	Trusted Advisor
CloudTrail	EC2	Inspector	RDS	VPC
CloudWatch Application Insights	EC2 - Fleet	IoT	Redshift	WorkLink
CloudWatch Events	EC2 Auto Scaling	IoT Things Graph	Rekognition	WorkMail
CodeBuild	EKS			

Select your use case

EKS

Allows EKS to manage clusters on your behalf.

EKS - Fargate pod

Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.

EKS - Fargate profile

Allows EKS to run Fargate tasks.

EKS - Nodegroup

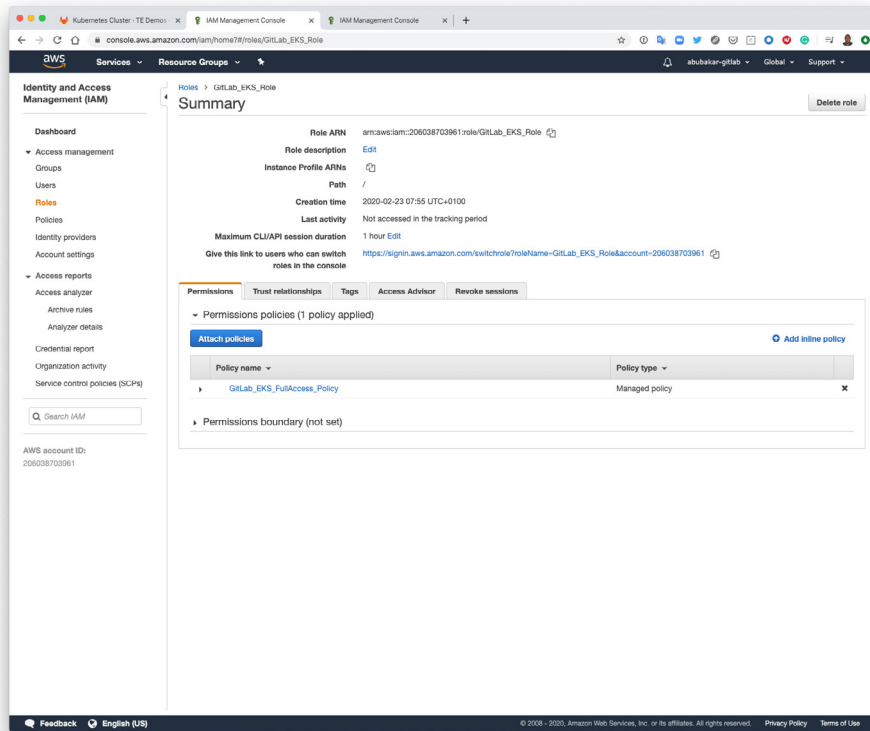
Allow EKS to manage nodegroups on your behalf.

* Required

Cancel

Next: Permissions

- You will notice the “AmazonEKSClusterPolicy” and “AmazonEKSServicePolicy” permissions are selected. Click through the Tags step and create if necessary, then click Next to get to the Review step. Click “Create Role” to complete the process.



GitLab EKS Integration

You only need to create the Provision and Service role once if you don't already have them in your organization's AWS setup. You can reuse the roles for other integrations or cluster creations.

1. Return to the GitLab Kubernetes Integration page and provide the Role ARN of the Provision Role we created earlier and click “Authenticate with AWS.”

TE Demos > Kubernetes

Add a Kubernetes cluster integration


Adding a Kubernetes cluster to your group will automatically share the cluster across all your projects. Use review apps, deploy your applications, and easily run your pipelines for all projects using the same cluster.


[Learn more about group Kubernetes clusters](#)

Create new cluster on EKS

Add existing cluster

Create cluster on


Amazon EKS


Google GKE

Authenticate with Amazon Web Services

You must grant access to your organization's AWS resources in order to create a new EKS cluster. To grant access, create a provision role using the account and external ID below and provide us the ARN.

Account ID

External ID

855262394183

94a9f5703b0c7d7b31b16c8a28843c78b38b6102

Create a provision role on [Amazon Web Services](#) using the account and external ID above. [More information](#)

Provision Role ARN

The Amazon Resource Name (ARN) associated with your role. If you do not have a provision role, first create one on [Amazon Web Services](#) using the above account and external IDs. [More information](#)

Authenticate with AWS

2. Once authenticated, you'll have a page to set the parameters needed to set up your cluster as shown in the image below and click on “Create Kubernetes Cluster” to let GitLab do its magic!

The parameters you'll need to provide are:

- » **Kubernetes cluster name** - The name you wish to give the cluster.
- » **Environment scope** - The GitLab environment associated with this cluster; * denotes the cluster will be used for deployments to all environments.
- » **Kubernetes version** - The Kubernetes version to use. Currently, the only version supported is 1.14.
- » **Role name** - The service role we created earlier.
- » **Region** - The AWS region in which the cluster will be created.
- » **Key pair name** - Select the key pair that you can use to connect to your worker nodes if required.
- » **VPC** - Select a VPC to use for your EKS Cluster resources.
- » **Subnets** - Choose the subnets in your VPC where your worker nodes will run.

- » **Security group** - Choose the security group to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets. AWS provides a default group, which can be used for the purpose of this guide. However, you are advised to setup up the right rules required for your resources.
- » **Instance type** - The AWS instance type of your worker nodes.
- » **Node count** - The number of worker nodes.
- » **GitLab-managed cluster** - Leave this checked if you want GitLab to manage namespaces and service accounts for this cluster.

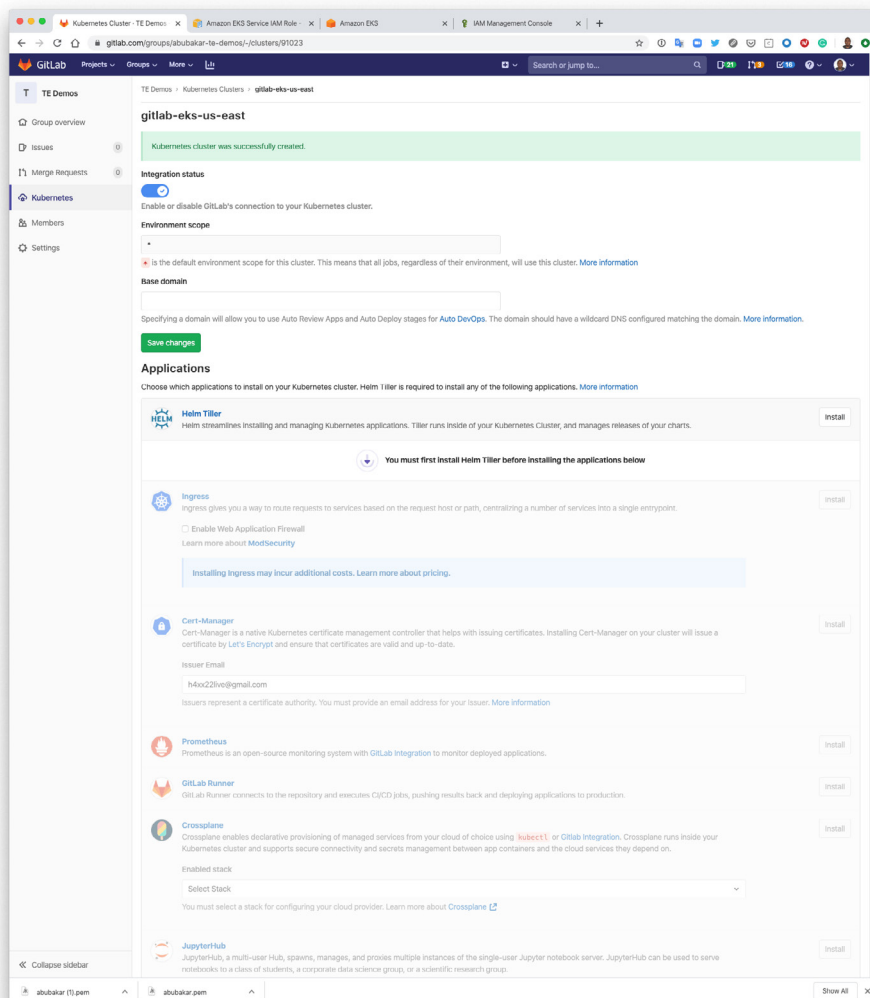
The screenshot shows the GitLab web interface for adding a new Kubernetes cluster. The left sidebar shows the 'Kubernetes' section under 'TE Demos'. The main content area is titled 'Add a Kubernetes cluster integration' and provides instructions on how to add a cluster. Below this, there are two tabs: 'Create new cluster on EKS' (selected) and 'Add existing cluster'. The 'Create new cluster on EKS' tab shows a form to enter details for the cluster. The form includes the following fields and options:

- Create cluster on:** Two buttons for 'Amazon EKS' and 'Google GKE'.
- Enter the details for your Amazon EKS Kubernetes cluster:**
 - Kubernetes cluster name:** A text input field containing 'gitlab-eks-us-east'.
 - Environment scope:** A dropdown menu with a single option '•'.
 - Kubernetes version:** A dropdown menu with the option '1.14'.
 - Service role:** A dropdown menu with the option 'GitLab_EKS_Service_Role'.
 - Region:** A dropdown menu with the option 'us-east-2'.
 - Key pair name:** A dropdown menu with the option 'abubakar'.
 - VPC:** A dropdown menu with the option 'vpc-a4ef20cf'.
 - Subnets:** A dropdown menu with the option 'subnet-d9db4f95, subnet-7d38cf16, subnet-2fd3fe55'.
 - Security group:** A dropdown menu with the option 'default'.
 - Instance type:** A dropdown menu with the option 't2.large'.
 - Number of nodes:** A text input field containing '3'.
 - GitLab-managed cluster:** A checked checkbox.

At the bottom of the form, there is a green button labeled 'Create Kubernetes cluster'.

3. The cluster creation process will take approximately 10 minutes. Once done you can proceed to install some predefined applications. At the very least, you need to install the following:

- » **Helm Tiller:** This is required to install the other applications.
- » **Ingress:** This provides SSL termination, load balancing and name-based virtual hosting you your applications. It acts as a web proxy for your application, which is useful when using AutoDevOps or deploying your own apps.
- » **Cert Manager:** This is a native Kubernetes certificate management controller, which helps in issuing certificates using Let's Encrypt. You don't need this if you want to use a custom Certificate issuer.
- » **Prometheus:** GitLab uses the Prometheus integration for automatic monitoring of your applications to collect metrics from Kubernetes containers allowing you to understand what is going on from within the GitLab UI.



3. To make use of Auto Review Apps and Auto Deploy stages of [AutoDevOps](#), you will need to specify a Base Domain name with a wild card DNS pointing to the Ingress Endpoint generated when you Install Ingress in the list of predefined apps

Additional resources:

- » [Adding and removing Kubernetes clusters](#)
- » [GitLab 12.5 with EKS Cluster Creation & Environments Dashboard](#)
- » [Preparing EKS resources](#)

GitLab + ECS

[Amazon Elastic Container Service \(ECS\)](#) is a fully managed container orchestration service. ECS integrates with many other services, such as AWS Identity and Access Management (IAM), and Amazon CloudWatch. ECS also gives applications the flexibility to use a mix of Amazon EC2 and AWS Fargate with Spot and On-Demand pricing options.

GitLab provides a series of [CI templates that you can include in your project](#). To automate deployments of your application to your Amazon Elastic Container Service (ECS) cluster, you can include the `Deploy-ECS.gitlab-ci.yml` template in your `.gitlab-ci.yml` file.

Before getting started with this process, you will need a cluster on AWS ECS, as well as related components, like an ECS service, ECS task definition, a database on AWS RDS, etc.

After you're all set up on AWS ECS, follow these steps:

1. Make sure your AWS credentials are set up as environment variables for your project. You can follow the [steps above](#) to complete this setup.
2. Add these variables to your project's `.gitlab-ci.yml` file:

» **variables:**

- » `CI_AWS_ECS_CLUSTER: my-cluster`
- » `CI_AWS_ECS_SERVICE: my-service`
- » `CI_AWS_ECS_TASK_DEFINITION: my-task-definition`

» **Three variables are defined in this snippet:**

- » CI_AWS_ECS_CLUSTER: The name of your AWS ECS cluster that you're targeting for your deployments.
- » CI_AWS_ECS_SERVICE: The name of the targeted service tied to your AWS ECS cluster.
- » CI_AWS_ECS_TASK_DEFINITION: The name of the task definition tied to the service mentioned above.

You can find these names after selecting the targeted cluster on your AWS ECS dashboard:

Clusters > my-cluster

Cluster : my-cluster Update Cluster Delete Cluster

Get a detailed view of the resources on your cluster.

Status **ACTIVE**

Registered container instances 1

Pending tasks count 0 Fargate, 0 EC2

Running tasks count 0 Fargate, 1 EC2

Active service count 0 Fargate, 1 EC2

Draining service count 0 Fargate, 0 EC2

Services Tasks ECS Instances Metrics Scheduled Tasks Tags Capacity Providers

Create Update Delete Actions Last updated on March 4, 2020 3:42:39 PM (2m ago)

Filter in this page Launch type ALL Service type ALL < 1-1 >

Service Name	Status	Service type	Task Definition	Desired tasks	Running tasks	Launch type	Platform version
my-service	ACTIVE	REPLICA	my-task-definition:10	1	0	EC2	--

3. Include this template in .gitlab-ci.yml:

```
include:  
  - template: Deploy-ECS.gitlab-ci.yml
```

4. The Deploy-ECS template ships with GitLab and is available on GitLab.com.

Commit and push your updated .gitlab-ci.yml to your project's repository, and you're done!

The application Docker image will be rebuilt and pushed to the GitLab registry. The targeted task definition will be updated with the location of the new Docker image, and a new revision will be created in ECS as result.

Last, your AWS ECS service will be updated with the new revision of the task definition, making the cluster pull the newest version of your application.

Additional resources:

- » [Deploy your application to AWS Elastic Container Service \(ECS\)](#)
- » [Automatically deploy to ECS \(Amazon Container Service\) \(#39089\) · Issues · GitLab.org](#)
- » [How to configure a Gitlab / ECS continuous deployment pipeline](#)

GitLab + EC2

Amazon Elastic Compute Cloud (EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

If you would like to connect EC2 and GitLab from the AWS Console, select an Amazon Machine Image (AMI) from the available templates.

Choose an instance type optimized to fit your unique use cases:

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how to choose.

Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECU, 1 vCPU, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs (1)	Memory (GiB)	Instance Storage (GiB) (1)	EBS-Optimized Available (1)	Network Performance (1)
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
<input checked="" type="checkbox"/>	General purpose	t2.micro	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate
<input type="checkbox"/>	General purpose	t2.xlarge	8	32	EBS only	-	Moderate
<input type="checkbox"/>	General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3a.micro	2	1	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3a.small	2	2	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3a.medium	2	4	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3a.large	2	8	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3a.xlarge	4	16	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3a.2xlarge	8	32	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3.small	2	2	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3.medium	2	4	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3.large	2	8	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3.xlarge	4	16	EBS only	Yes	Up to 5 Gigabit
<input type="checkbox"/>	General purpose	t3.2xlarge	8	32	EBS only	Yes	Up to 5 Gigabit

3:27

Cancel Previous Review and Launch

Your GitLab instance is going to need to talk to the Runners over the network, and that is something you need to think about when configuring any AWS security groups or when setting up your DNS configuration.

Developers can also use [Terraform](#) to spin up an AWS instance to easily target EC2. If you would like more information, we've included [a sample Terraform project](#) to launch on AWS EC2.

GitLab can be a useful tool for utilizing EC2 spot instances. Amazon EC2 Spot instances allow you to bid on spare Amazon EC2 computing capacity. Since Spot instances are often available at a discount compared to On-Demand pricing, you can significantly reduce the cost of running your applications, grow your application's compute capacity and throughput for the same budget, and enable new types of cloud computing applications.

Substrakt Health, a technology company that provides digital solutions for the National Health Service (NHS), used [Autoscaling GitLab Runners](#) to save 90% on their EC2 costs.

[Read their story](#)

Additional resources:

- » [Creating an IAM EC2 instance role and profile](#)
- » [Deploy Code to AWS EC2 Instances Automatically with GitLab](#)
- » [Autoscaling GitLab Runner on AWS EC2](#)

Additional AWS functionality in the works

- » 12.10 [Autoscaling GitLab CI jobs on AWS Fargate](#)
- » 13.0 [Template for Deploying to AWS EC2](#)
- » 13.1 [Offer in-product guidance for deploying to AWS](#)

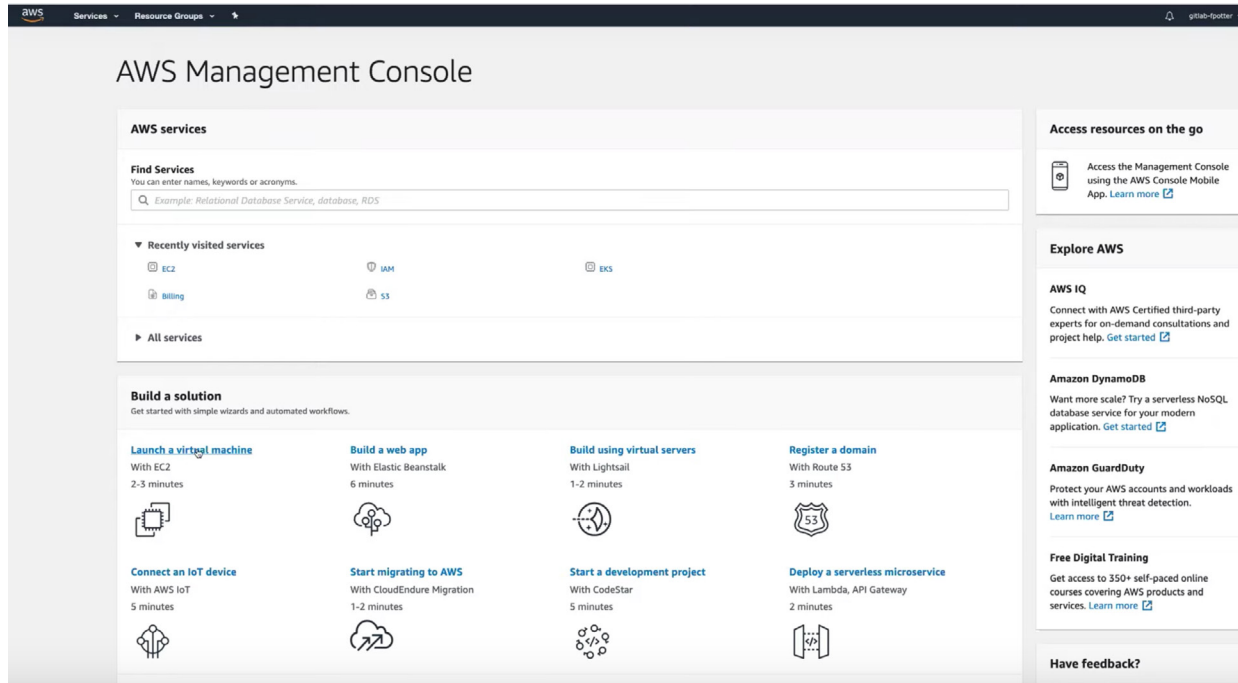
GitLab releases new features on the 22nd of every month. For a list of release posts including patch releases, please check the blog category [releases](#). Future releases, and their important features, can be found on our [upcoming releases](#) page. You can also view [upcoming features by product tier](#).

Getting started with GitLab

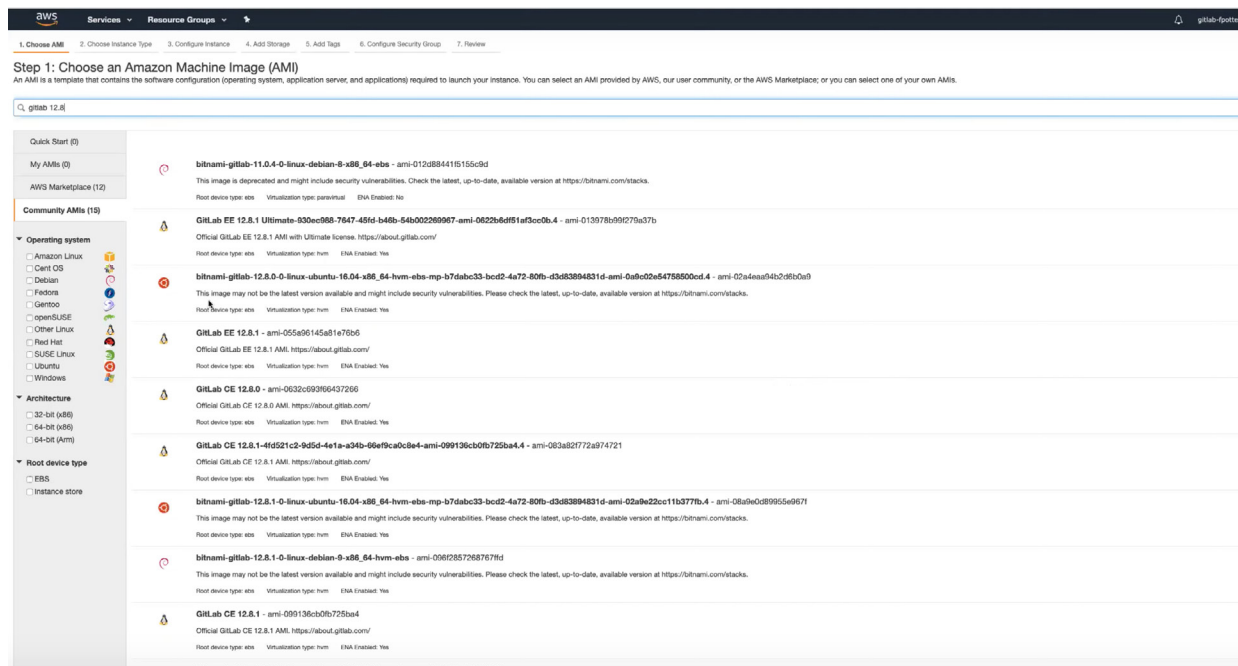
Changing DevOps processes can be an organizational challenge, so a proof of concept is essential for evaluating feasibility. Every team has different capabilities and priorities, and new processes should be properly vetted. Luckily, teams looking to try GitLab for their AWS deployments have a few options.

Installing GitLab from AWS Management console

Go the [AWS Management Console](#) and select **LAUNCH A VIRTUAL MACHINE**



Next, click on **COMMUNITY AMIS** and search for the latest version of GitLab (ex: GitLab 12.8, GitLab 12.9 etc). Click **SELECT**



At the **CHOOSE AN INSTANCE TYPE**, select your instance. GitLab recommends **T2.MEDIUM** or higher

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they work.

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (variable ECU, 1 vCPU, 2.5 Gbit, Intel Xeon Family, 1 GiB memory, EBS only)

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
General purpose	t2.micro	1	1	EBS only	-	Low to Moderate
General purpose	t2.small	1	2	EBS only	-	Low to Moderate
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
General purpose	t2.large	2	8	EBS only	-	Low to Moderate
General purpose	t2.xlarge	4	16	EBS only	-	Moderate
General purpose	t2.2xlarge	8	32	EBS only	-	Moderate
General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gbit
General purpose	t3a.micro	2	1	EBS only	Yes	Up to 5 Gbit
General purpose	t3a.small	2	2	EBS only	Yes	Up to 5 Gbit
General purpose	t3a.medium	2	4	EBS only	Yes	Up to 5 Gbit
General purpose	t3a.large	2	8	EBS only	Yes	Up to 5 Gbit
General purpose	t3a.xlarge	4	16	EBS only	Yes	Up to 5 Gbit
General purpose	t3a.2xlarge	8	32	EBS only	Yes	Up to 5 Gbit
General purpose	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gbit
General purpose	t3.micro	2	1	EBS only	Yes	Up to 5 Gbit
General purpose	t3.small	2	2	EBS only	Yes	Up to 5 Gbit
General purpose	t3.medium	2	4	EBS only	Yes	Up to 5 Gbit
General purpose	t3.large	2	8	EBS only	Yes	Up to 5 Gbit
General purpose	t3.xlarge	4	16	EBS only	Yes	Up to 5 Gbit
General purpose	t3.2xlarge	8	32	EBS only	Yes	Up to 5 Gbit

3:27

Select an existing key pair or create a new key pair and select **LAUNCH INSTANCES**

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process.

Your instance configuration is not eligible for the free usage tier. To launch an instance that's eligible for the free usage tier, check your AMI selection, instance type, configuration options, or storage devices. [Learn more about free usage tier eligibility and usage restrictions.](#)

AMI Details

GitLab EE 12.8.1 - ami-055a96145d1e70b6
Official GitLab EE 12.8.1 AMI. <https://about.gitlab.com/>
Root Device Type: x86_64

Instance Type

Instance Type	ECU	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized
t2.medium	Variable	2	4	EBS only	-

Security Groups

Security group name: launch-wizard-6
Description: launch-wizard-6 created 2020-02-27T13:51:16.345-08:00

Type: | Protocol: | Port Range: | Source: |

This security group has no rules.

Instance Details

Storage

Tags

Select an existing key pair or create a new key pair

A key pair consists of a public key that AWS stores, and a private key file that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. [Learn more about removing existing key pairs from a public AMI.](#)

Choose an existing key pair

Select a key pair

gitlab-awsnet-1-keypair

☐ I acknowledge that I have access to the selected private key file (gitlab-awsnet-1-keypair.pem), and that without this file, I won't be able to log into my instance.

Cancel Launch Instances

You're done! You should now be able to access your GitLab Community Edition instance from the browser. If it does not appear, make sure to check the security group.

The screenshot shows the AWS Management Console interface. On the left, there's a navigation menu with categories like EC2 Dashboard, INSTANCES, IMAGES, ELASTIC BLOCK STORE, NETWORK & SECURITY, LOAD BALANCING, and AUTO SCALING. The main area displays a list of EC2 instances. A dropdown menu is open for the instance 'i-046ea9707b846d75', showing options like 'Connect', 'Create Snapshot From Instance', 'Launch More Like This', 'Instance State', 'Instance Settings', 'Image', 'Networking', 'CloudWatch Monitoring', 'Check Security Groups', 'Attach Network Interface', 'Detach Network Interface', 'Change Source/Target Check', and 'Manage IP Addresses'. Below the list, the details for instance 'i-046ea9707b846d75' are shown, including its Public DNS, IP address, and various configuration details.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	IPv4 Public IP	Key Name	Monitoring	Launch Time	Security Groups
kubeflow-cluster-kubeflow-cluster-...	i-046ea9707b846d75	t2.medium	us-east-1c	running	2/2 checks ...	None	54.145.48.96	gitlab-usesat1-...	disabled	February 27, 2020 at 1:51:30 PM UTC-8	launch-wizard-6
kubeflow-cluster-kubeflow-cluster-...	i-0502168595b...	t3.medium	us-east-1c	running	2/2 checks ...	None	34.229.17.180	gitlab-usesat1-...	enabled	November 21, 2019 at 4:17:...	kubeflow-nodestack-NodeSecurityGroup-7C
kubeflow-cluster-kubeflow-cluster-...	i-0502168595b...	t3.medium	us-east-1b	running	2/2 checks ...	None	54.152.185.166	gitlab-usesat1-...	enabled	November 21, 2019 at 4:17:...	kubeflow-nodestack-NodeSecurityGroup-7C
kubeflow-cluster-kubeflow-cluster-...	i-0502168595b...	t3.medium	us-east-1a	running	2/2 checks ...	None	34.205.87.91	gitlab-usesat1-...	enabled	November 21, 2019 at 4:17:...	kubeflow-nodestack-NodeSecurityGroup-7C
kubeflow-cluster-kubeflow-cluster-...	i-0502168595b...	t3.medium	us-east-1d	running	2/2 checks ...	None	54.182.181.94	gitlab-usesat1-...	disabled	October 7, 2019 at 12:49:22 ...	web-secgroup

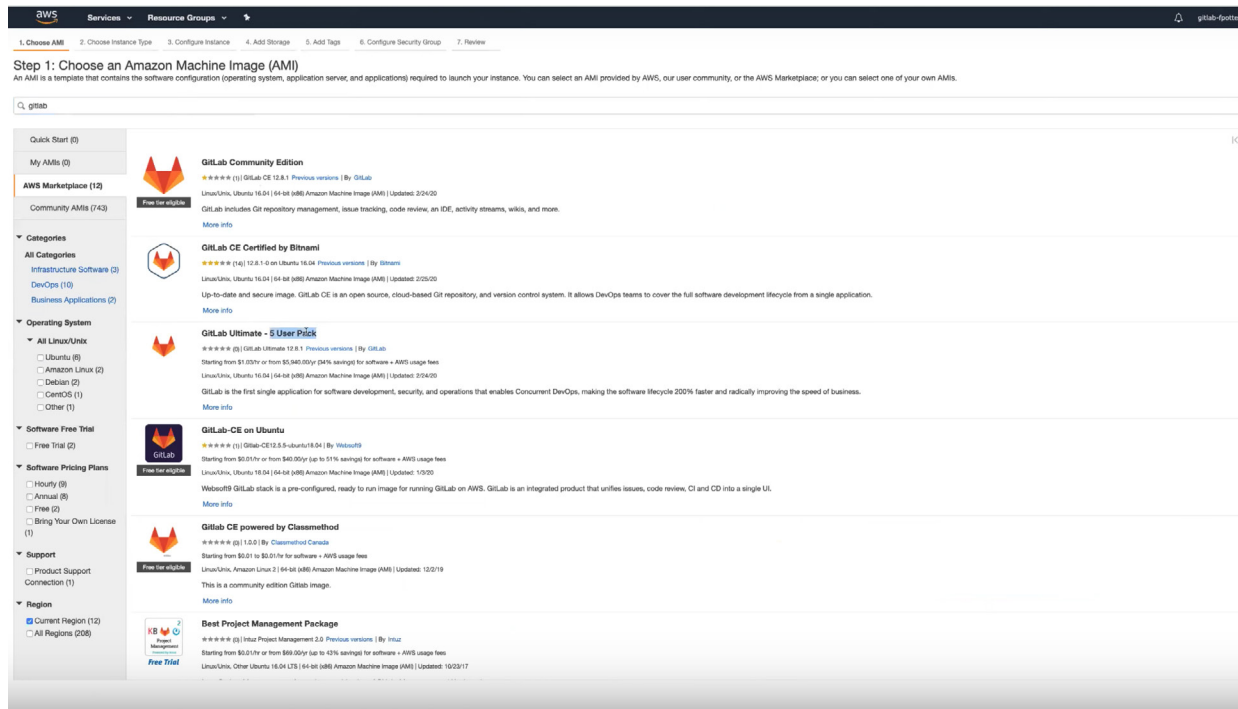
Instance: **i-046ea9707b846d75** Public DNS: ec2-54-145-48-96.compute-1.amazonaws.com

Description	Status Checks	Monitoring	Tags
Instance ID	i-046ea9707b846d75		
Instance state	running		
Instance type	t2.medium		
Findings	Opt-in to AWS Compute Optimizer for recommendations. Learn more		
Private DNS	ip-172-31-24-78.ec2.internal		
Private IP	172.31.24.78		
Secondary private IP			
VPC ID	vpc-c6e1150d:us-east-1		
Subnet ID	subnet-9c3baa8f:us-east-1		
Network interfaces	eni0		
Source/dest. check	True		
T2/T3 Unlimited	Disabled		
FBI5-optimized	False		
Root device type	ebs		

Public DNS (IPv4) ec2-54-145-48-96.compute-1.amazonaws.com
IPv4 Public IP 54.145.48.96
IPv4 IPs -
Elastic IPs -
Availability zone us-east-1c
Security groups launch-wizard-6. [View inbound rules.](#) [View outbound rules.](#)
Scheduled events No scheduled events
AMI ID GitLab EE 12.8.1 (ami-055a8f145a81a7b04)
Platform -
IAM role -
Key pair name gitlab-usesat1-keypair
Owner 244747684470
Launch time February 27, 2020 at 1:51:30 PM UTC-8 (less than one hour)
Termination protection False

Installing GitLab from AWS Marketplace

Another option is going through the [AWS Marketplace](#). When you search for GitLab, select the **GITLAB ULTIMATE - 5 USER PACK** option. Finish the setup prompts and you're done.

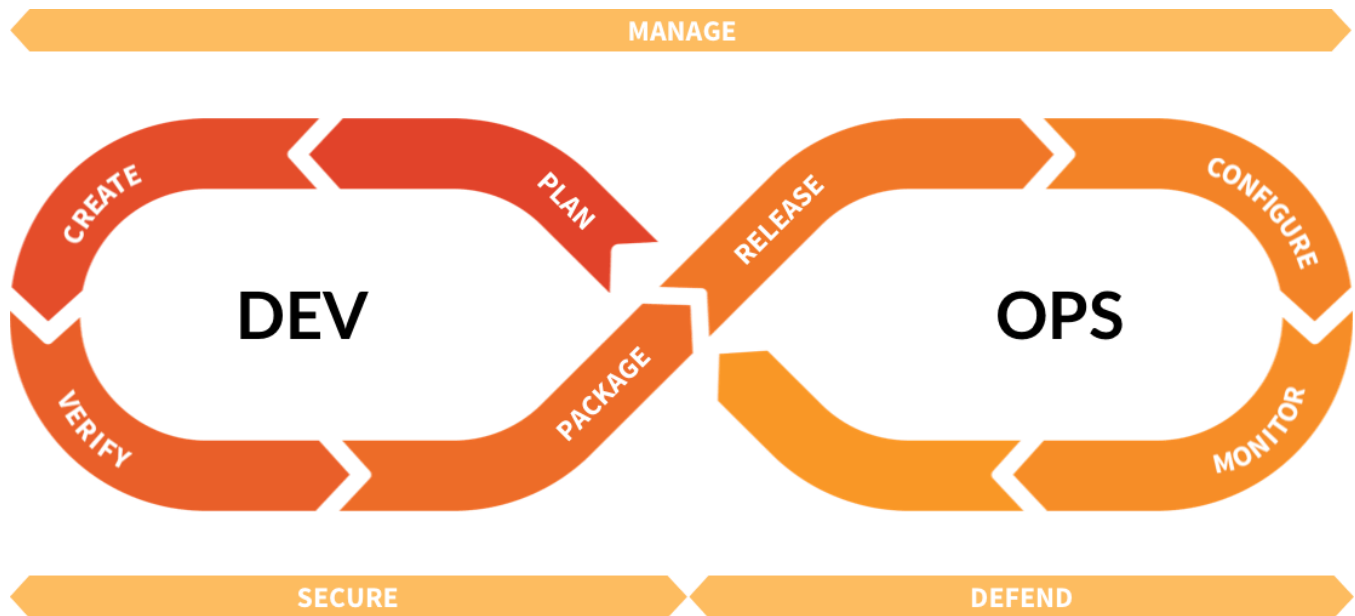


The screenshot shows the AWS Marketplace search results for "gitlab". The left sidebar contains navigation options: Quick Start (0), My AMIs (0), AWS Marketplace (12), and Community AMIs (743). Under "Categories", there are links for Infrastructure Software (2), DevOps (10), and Business Applications (2). The "Operating System" section lists "All Linux/Unix" with sub-options for Ubuntu (8), Amazon Linux (2), Debian (2), CentOS (1), and Other (1). The "Software Free Trial" section shows "Free Trial (2)". The "Software Pricing Plans" section lists "Hourly (9)", "Annual (8)", "Free (2)", and "Bring Your Own License (1)". The "Support" section shows "Product Support Connection (1)". The "Region" section shows "Current Region (12)" and "All Regions (208)".

The main content area displays search results for "gitlab". The first result is "GitLab Community Edition" (4.5 stars, 11 reviews), which is a free trial. The second result is "GitLab CE Certified by Bitnami" (4.5 stars, 14 reviews), which is a free trial. The third result is "GitLab Ultimate - 5 User Pack" (4.5 stars, 2 reviews), which is a paid option starting at \$1,037/hr or \$5,940.00/yr. The fourth result is "GitLab CE on Ubuntu" (4.5 stars, 11 reviews), which is a free trial. The fifth result is "GitLab CE powered by Classroom" (4.5 stars, 2 reviews), which is a free trial. The sixth result is "Best Project Management Package" (4.5 stars, 2 reviews), which is a free trial.

Note: the GitLab Ultimate - 5 User Pack is not a free trial on the AWS Marketplace, but you will receive a GitLab license, as well as access to our [Ultimate](#) features. This is a great option if you would like to try the full GitLab experience with a limited number of users first.

Benefits of using GitLab for AWS services



Organizations love AWS because it serves as a one-stop-shop for all cloud computing and IT needs. An all-in-one cloud platform can improve efficiency by bringing visibility into cloud processes and cloud usage. DevOps tools can make a big impact on cloud usage, but If development is bogged down by brittle toolchains and inefficient DevOps processes, teams won't be able to manage resources effectively.

GitLab accelerates software delivery by driving higher efficiency across all stages of the software development lifecycle. Increased visibility means teams are more productive, reducing handoffs and eliminating bottlenecks. Development, Security, and Ops teams can collaborate in one interface and deploy to any AWS infrastructure.

All-in-one cloud meets all-in-one DevOps

By running GitLab on AWS you get a complete DevOps platform running and deploying to AWS cloud infrastructure. GitLab and AWS offer similar benefits to organizations looking to streamline their development process.

- » **Better visibility:** Teams using AWS can utilize multiple cloud services on a single platform, and DevOps teams can build, stage and deploy to the services from one application using GitLab.
- » **Increased efficiency:** All-in-one cloud means teams don't have to manage multiple cloud platforms, and GitLab's single interface means teams can collaborate across all stages of the SDLC.
- » **Reduced cycle times:** Instead of managing multiple clouds, processes, and tools, teams can utilize AWS and GitLab and focus on building software (not maintenance).

As organizations continue their journey around digital transformation, DevOps has become the go-to set of best practices to increase their velocity of delivering value to the business while increasing quality. Breaking down the walls and reducing the handoffs necessary to complete a full delivery lifecycle is also very important and remains a challenge to most organizations.

GitLab is a complete DevOps platform, delivered as a single application with bring-your-own-infrastructure flexibility. By running GitLab on AWS you get a complete DevOps platform, delivered as a single application, running and deploying to your AWS cloud infrastructure.

[Sign up for a 30-day free trial](#)

About GitLab

GitLab is a DevOps platform built from the ground up as a single application for all stages of the DevOps lifecycle enabling Product, Development, QA, Security, and Operations teams to work concurrently on the same project.

GitLab provides teams a single data store, one user interface, and one permission model across the DevOps lifecycle allowing teams to collaborate and work on a project from a single conversation, significantly reducing cycle time and focus exclusively on building great software quickly.

Built on Open Source, GitLab leverages the community contributions of thousands of developers and millions of users to continuously deliver new DevOps innovations. More than 100,000 organizations from startups to global enterprise organizations, including Ticketmaster, Jaguar Land Rover, NASDAQ, Dish Network and Comcast trust GitLab to deliver great software at new speeds. GitLab is the world's largest all-remote company, with more than 1,200 team members in over 65 countries.

