Tampereen yliopisto
Tampere University

Lecture 09

Project; GitLab CI; Cloud Native – part 3

Kari Systä
27.10.2020

Tampereen yliopisto
Tampere University

Schedule for coming weeks

Week	Lecture	Plussa exercises (deadlines)
8/43	20.10 Cloud-native architectures part 2	19.10 Next exercise closes 25.10 Project instructions opens
9/44	27.10 Inroduction to project, Gitlab CI	
10/45	01.11 Testing and testing automation	
11/46	10.11 Guest Lecture, CD pipeline at cargotec	
12/47	17.11	
13/48	24.11	
14/40	01.12	

Master-thesis opportunities

- www.lansio.com
  - Math and SW for optimizing delivery work

- University and company collaboration
  - CO_2 footprint analysis application for energy systems

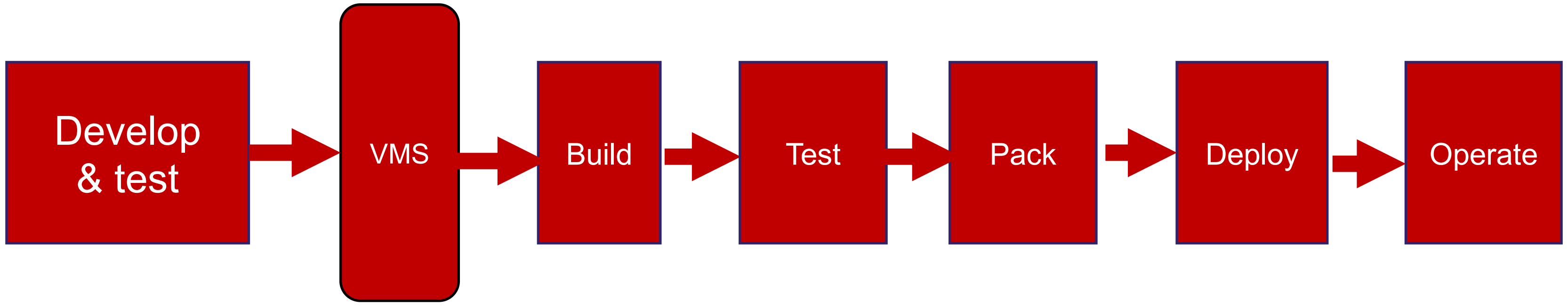
- University research project
  - Visualization of DevOps projects.

Course practicalities

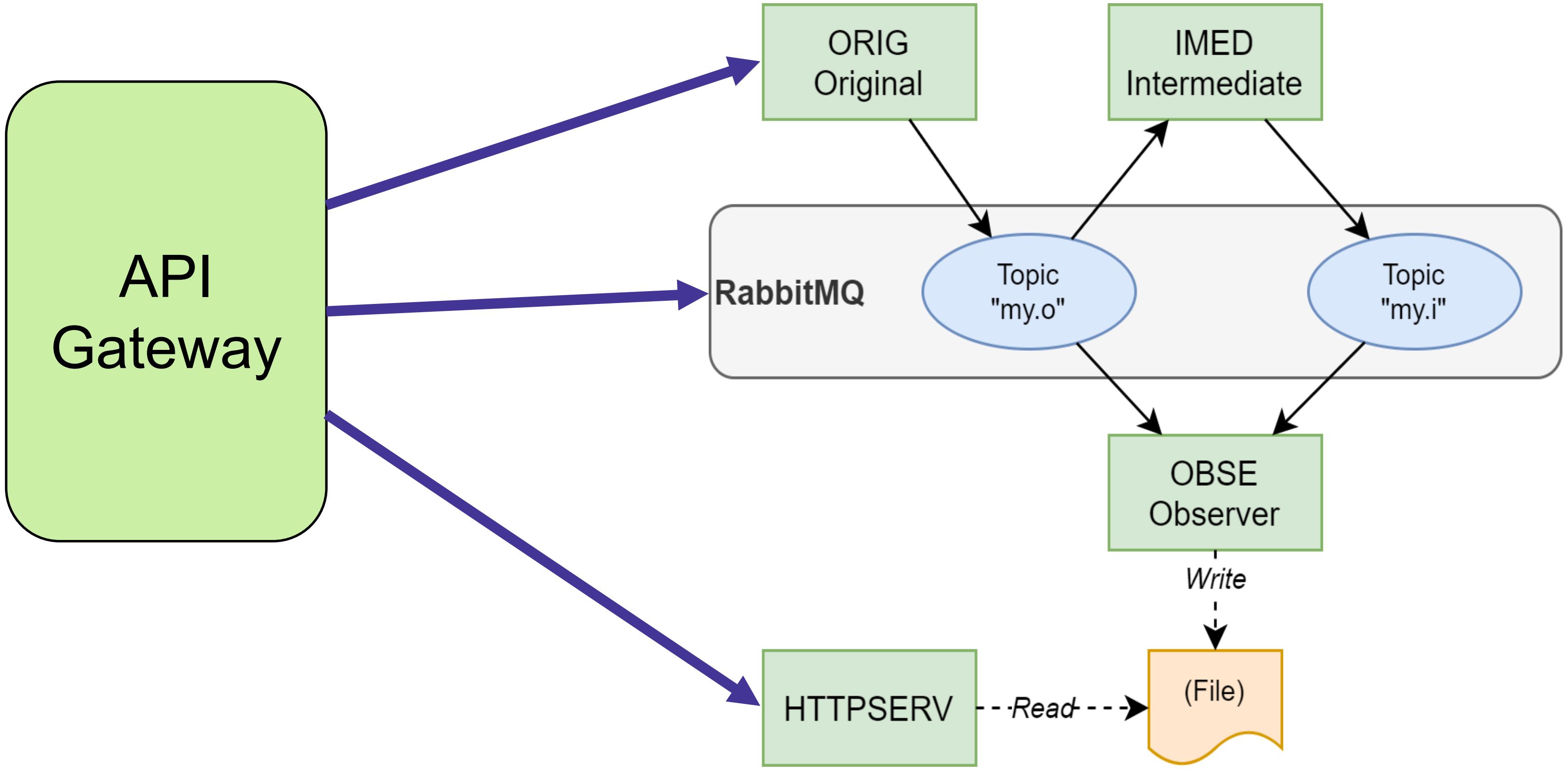
- Cloud exercise was returned by 70
- Compose exercise was returned by 52+4
- Message-queue communication has been returned by 50

- I have now checked most the compose-exercises and feedback given
  - Clarification requests will be handled later this week

Project

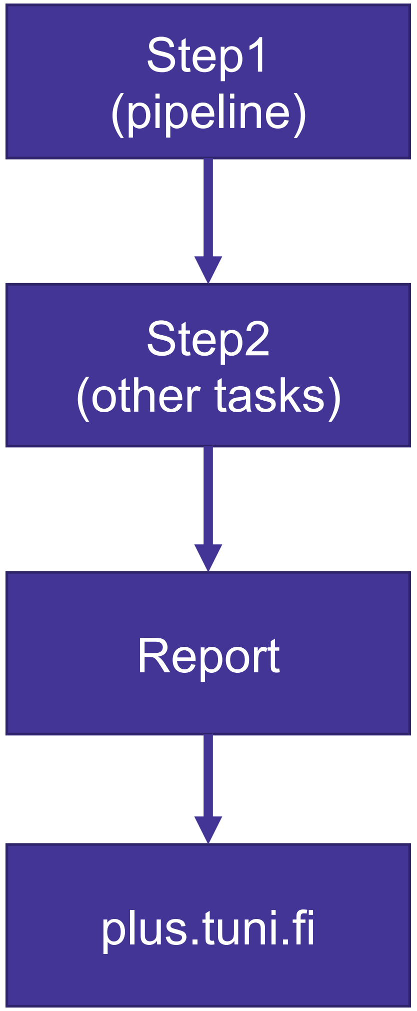


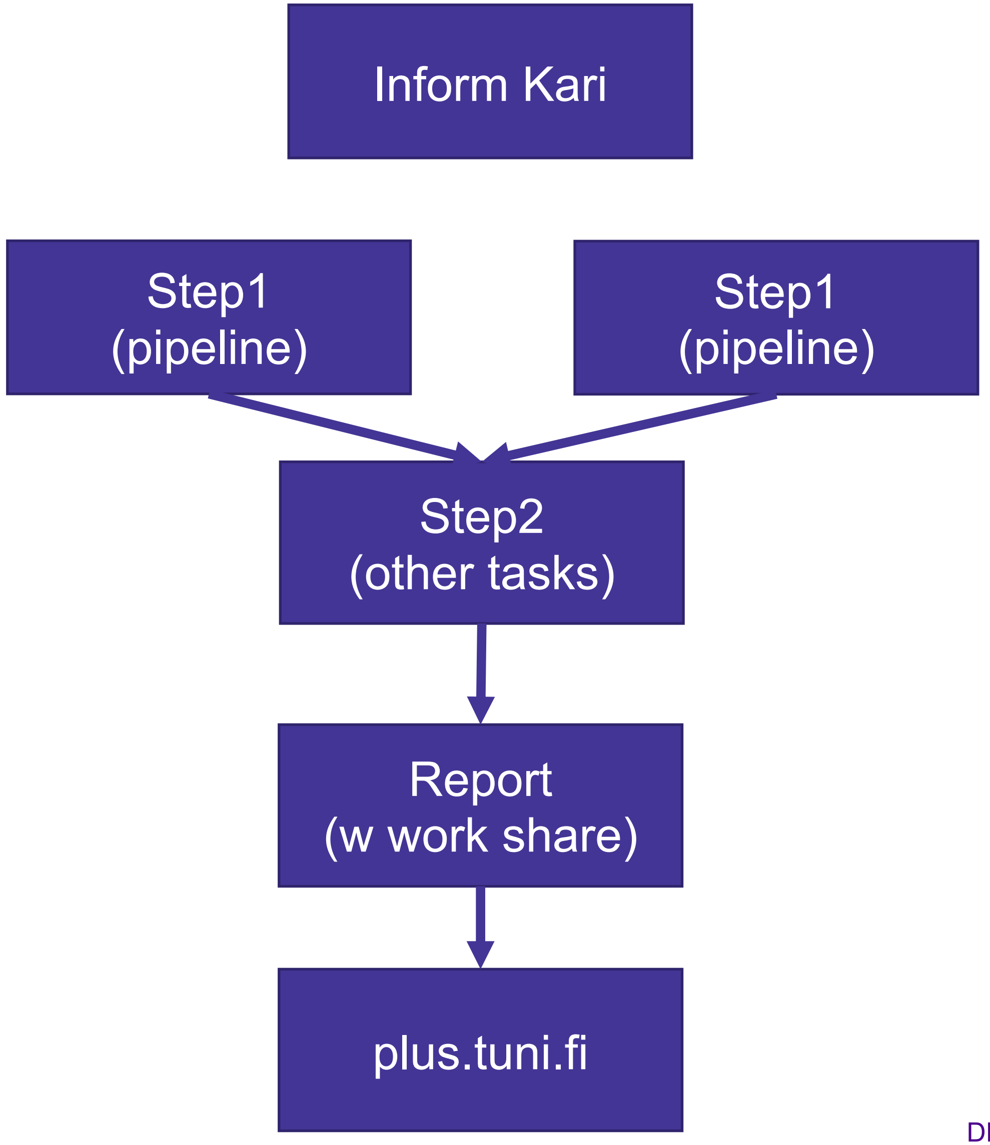
Develop & test
VMS
Build
Test
Pack
Deploy
Operate



API Gateway
ORIG Original
IMED Intermediate
RabbitMQ
Topic "my.o"
Topic "my.i"
OBSE Observer
Write
(File)
HTTPSERV
Read

Two options: individual or pair





Project includes

1. Install the pipeline infrastructure using gitlab-ci. This means that you should:
   - install gitlab and runners on their own machine. A fresh virtual machine is recommended. Instructions to help in this process are below in section gitlab-ci.
   - Define the pipeline using **`.gitlab-ci.yml`** for the application you implemented for the message-queue exercise. The result of the pipeline should be a running system, so the containers should be started automatically. (In other words: “git push => the system is up and running)
   - Test the pipeline with the current version of the application.
   - PAIR: if you are a member of a pair you should return the pipe-lines before entering the next phase. So, this first phase is still individual work (do not return equal pipelines!) even for pairs. Returning to be done with git URL in plussa.
2. Create, setup and test an automatic testing framework
   - First, you need to select the testing tools. We do not require any specific tool, even your own test scripts can be used.
   - Create test to the existing functionality of the application (see “Application and its new features” below)

Content of the project

3. Implements changes to the system by using the pipeline. The development should be done in test-driven manner (test before implementation – see https://en.wikipedia.org/wiki/Test-driven_development)
   - For each new feature, you should first implement tests, then implement the feature and after passing the tests move to next feature. This behavior should be verifiable from in the version history.
   - Tests mush be in a separate folder "tests" at the root of your folder tree.
4. Deploy the application at least to your own machine. *Optionally, deployment to external cloud (Heroku or similar).*
5. Modify the ORIG service to send messages forever until pause paused or stopped.
6. Implement an API gateway
7. Write the end report
8. *(Optional) implement a static analysis step in the pipeline by using tools like jlint, pylint or SonarQube.*
9. *(Optional) implement monitoring and logging for troubleshooting. This should be a separate service that the user can use through browser. It should show at least start time of the service, number of requests it has received after start.*

API gateway

GET /messages

Returns all message registered with OBSE-service

PUT /state (payload “INIT”, “PAUSED”, “RUNNING”, “SHUTDOWN”)

PAUSED = ORIG service is not sending messages

RUNNING = ORIG service sends messages

If the new state is equal to previous nothing happens.

There are two special cases:

INIT = everything is in the initial state and ORIG starts sending again, state is set to RUNNING
SHUTDOWN = all containers are stopped

GET /state

get the value of state

GET /run-log

Get information about state changes

Example output:

2020-11-01T06:35:01.373Z: INIT
2020-11-01T06:40:01.373Z: PAUSED
2020-11-01T06:40:01.373Z: RUNNING

GET /message-log

Forward the request to HTTPSERV and return the result

GET /node-statistic (optional)

Return core statistics (the five (5) most important in your mind) of the RabbitMQ. (For getting the information see https://www.rabbitmq.com/monitoring.html)
Output should syntactically correct and intuitive JSON. E.g:
{ “fd_used”: 5, …}

GET /queue-statistic (optional)

Return a JSON array per your queue. For each queue return “message delivery rate”, “messages publishing rate”, “messages delivered recently”, “message published lately”. (For getting the information see https://www.rabbitmq.com/monitoring.html)

End report

- Description of the CI/CD pipeline.
- Instructions for examiner to test the system. Pay attention to optional features. This need to be in the README.md-file
- Example runs (some kind of log) of both failing test and passing. The students need to show how the pipeline works both in case of success and failure.
- Main learnings and worst difficulties (especially, if you think that something should have been done differently, describe it here)
- Amount effort (hours) used
- PAIR: description of the individual roles of both students

Grading

As already been communicated this project affects 40% of in the evaluation of the overall course. For that 40% we use the following table

• Compulsory parts work according to requirements	0..20 %

PAIR:

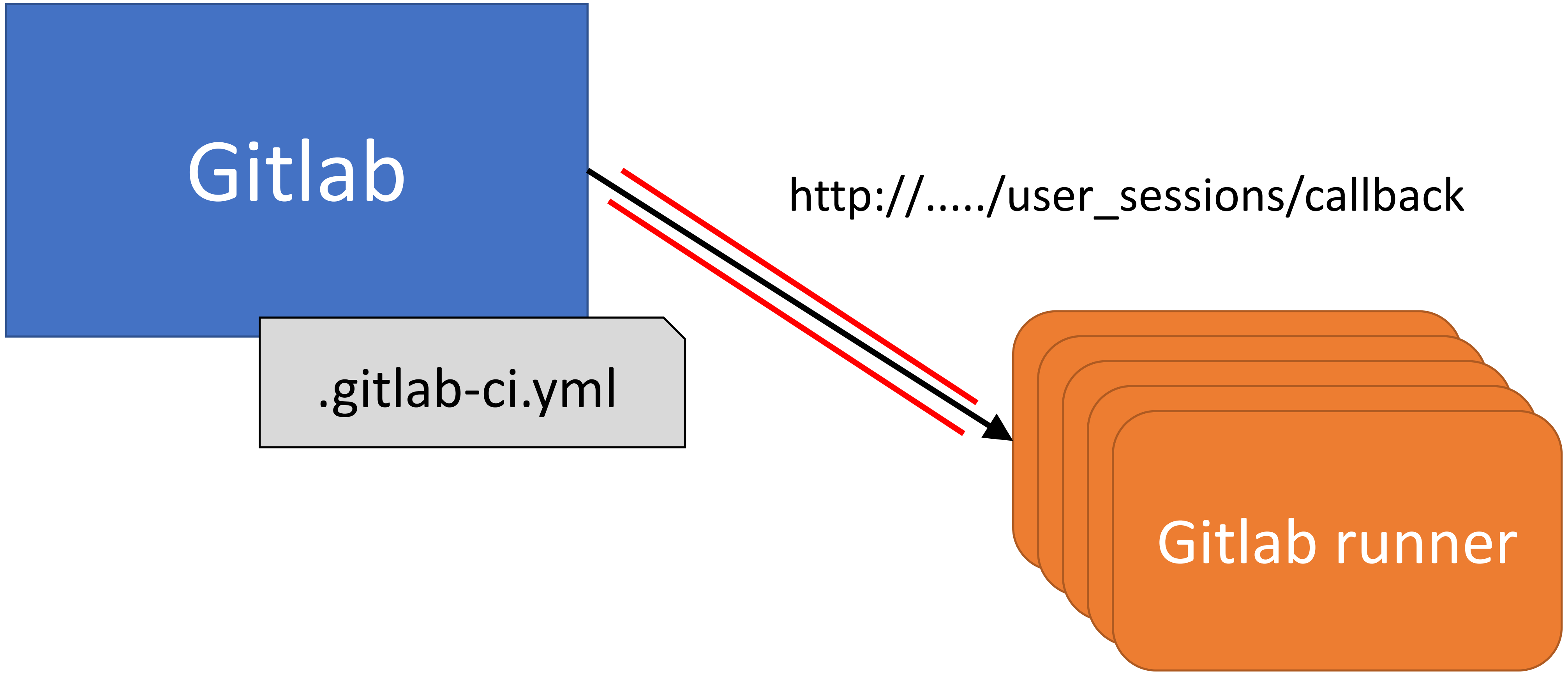
- at least one optional feature needs to be implemented to reach 25%
- pipelines are evaluated separately

Implementation of optional features	0..20 %
Overall quality (clean code, good comments, ….)	0..5%
Quality of the end report	0..5% (+ up to 5% compensation of a good analysis of your solution and description of a better way to implement.)

Tampereen yliopisto
Tampere University

Gitlab CI

https://docs.gitlab.com/ee/ci/



Types of runners

Shared Runners

- These runners are useful for jobs multiple projects which have similar requirements. Instead of using multiple runners for many projects, you can use a single or a small number of Runners to handle multiple projects which will be easy to maintain and update.

Specific Runners

- These runners are useful to deploy a certain project, if jobs have certain requirements or specific demand for the projects. Specific runners use *FIFO* (First In First Out) process for organizing the data with first-come first-served basis.

Tampereen yliopisto
Tampere University

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

Example from:
https://docs.gitlab.com/ee/user/project/pages/getting_started/pages_from_scratch.html

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

Base Image

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

This is run before every script

Tampereen yliopisto
Tampere University

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

Used rules

Many variables available: https://docs.gitlab.com/ee/ci/variables/predefined_variables.html

Use of rule, executed if rule is “master”

Tampereen yliopisto
Tampere University

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

This is for state ”deploy”.

Default states are build, test, deploy

This is for state ”test”.

```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

Never mind ☺

Script to run

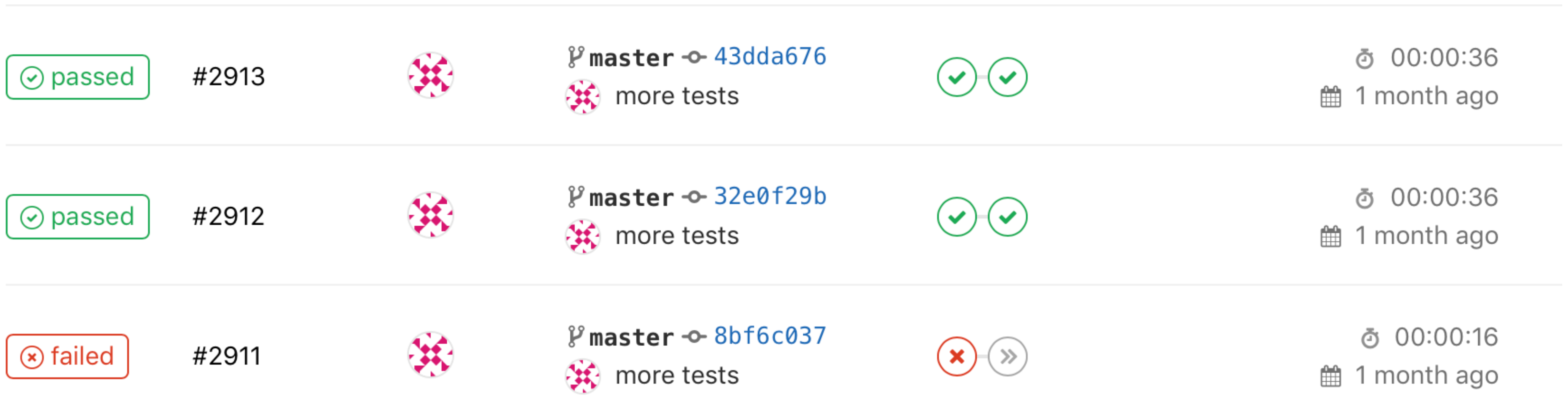
```
image: ruby:2.7

workflow:
  rules:
    - if: '$CI_COMMIT_BRANCH'

before_script:
  - gem install bundler
  - bundle install

pages:
  stage: deploy
  script:
    - bundle exec jekyll build -d public
  artifacts:
    paths:
      - public
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'

test:
  stage: test
  script:
    - bundle exec jekyll build -d test
  artifacts:
    paths:
      - test
  rules:
    - if: '$CI_COMMIT_BRANCH != "master"'
```

File location

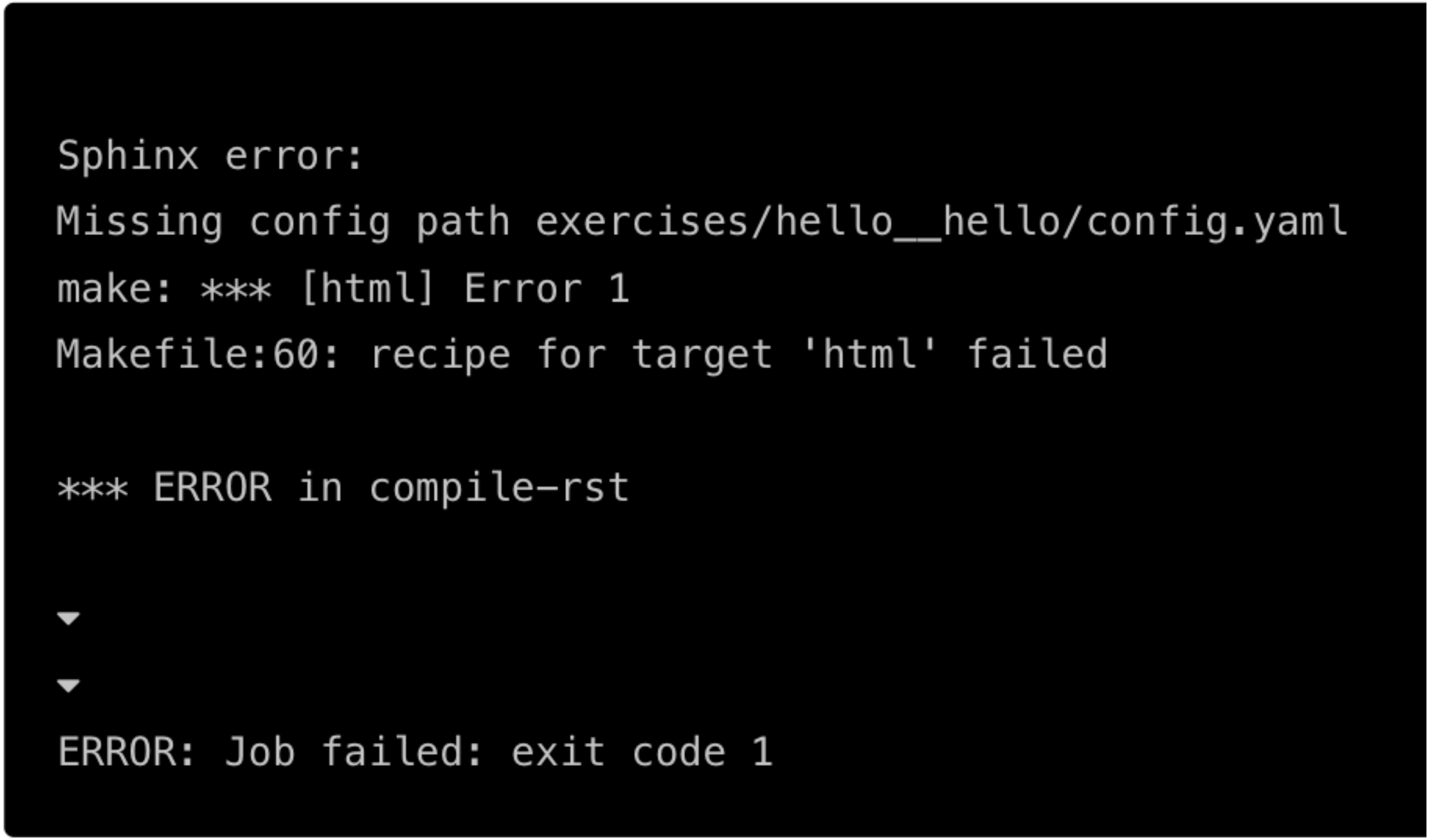
Tampereen yliopisto
Tampere University

How to install .gitlab-ci.yml?

```
git add .gitlab-ci.yml
git commit -m "Add .gitlab-ci.yml”
git push origin master
```



passed	#2913	master 43dda676 more tests	✓ ✓	00:00:36 1 month ago
passed	#2912	master 32e0f29b more tests	✓ ✓	00:00:36 1 month ago
failed	#2911	master 8bf6c037 more tests	✗ »	00:00:16 1 month ago



```
Sphinx error:
Missing config path exercises/hello__hello/config.yaml
make: *** [html] Error 1
Makefile:60: recipe for target 'html' failed

*** ERROR in compile-rst


▾

▾

ERROR: Job failed: exit code 1
```

```
variables:
    TUNIPLUSSA_ID: 'TIE23536-
syksy2019'
    GIT_STRATEGY: none

stages:
    - build
    - test
    - deploy

builder:
    stage: build
    only:
    - master
    - release
    tags:
    - plussa
    artifacts:
      paths:
      - FULLLOG.txt
      expire_in: 2 week
    script:
    - tuni-rst-build
```

```
tester:
    stage: test
    only:
    - master
    tags:
    - plussa
    script:
    - tuni-publish-to-testing


publisher:
    stage: deploy
    only:
    - release
    tags:
    - plussa
    script:
    - tuni-publish-to-production
```

```
variables:
    TUNIPLUSSA_ID: 'TIE23536-
syksy2019'
    GIT_STRATEGY: none

stages:
    - build
    - test
    - deploy

builder:
    stage: build
    only:
    - master
    - release
    tags:
    - plussa
    artifacts:
      paths:
      - FULLLOG.txt
      expire_in: 2 week
    script:
    - tuni-rst-build
```

```
tester:
    stage: test
    only:
    - master
    tags:
    - plussa
    script:
          ish-to-testing



          oy

    - release
    tags:
    - plussa
    script:
    - tuni-publish-to-production
```

Note: The rules syntax is an improved, more powerful solution for defining when jobs should run or not. Consider
using rules instead of only/except to get the most out of your pipelines.

Installing

- https://www.youtube.com/watch?v=yfsvaXubuUg

Using

- https://www.youtube.com/watch?v=Jav4vbUrqII

- https://docs.gitlab.com/ee/ci/quick_start/

Tampereen yliopisto
Tampere University

Function as a service/ serverless computing

Do you really want to keep
your containers running all the time
if you need to pay for it?

Do you really want to operate
and maintain your containers –
your developers could also
do something else.

Serverless computing

Baldini et all: Serverless Computing:

Current Trends and Open Problems, Research Advanges in Cloud Computing, Springer, 2017.

A cloud-native platform

for

- short-running, stateless computation
- event driven applications

which

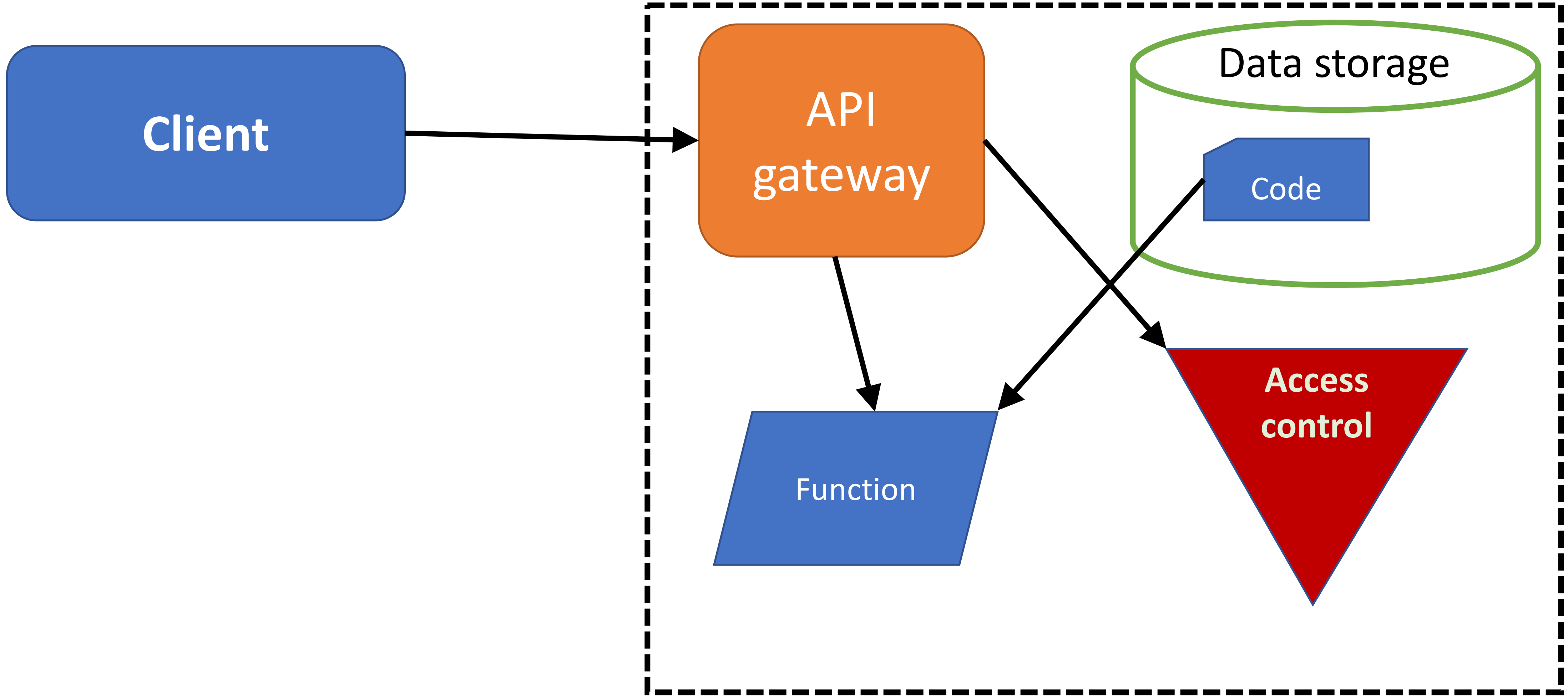
- scale up and down instantly and automatically

and

- charge for actual usage and high granulatity https://medium.com/@BoweiHan/an-introduction-to-serverless-and-faas-functions-as-a-service-fb5cec0417b2

”… you can simply upload modular chunks of functionality into the cloud that are executed independently.
Imagine the possibilities! Instead of scaling a monolithic REST server to handle potential load, you can now split the server into a bunch of functions which can be scaled automatically and independently.”

Function as a service?



Tampereen yliopisto
Tampere University

A simple example from https://www.scalyr.com/blog/simple-detailed-introduction-google-cloud-functions/

- Package.json

```
{ "name": "my-first-function", "version": "0.0.1" }
```

- Code

```
exports.helloWorld = (req, res) => {
    let message = req.query.message ||
        req.body.message || 'Hello World!’;
    res.status(200).send(message);
};
```

- Deploy with

```
gcloud functions deploy my-first-function --trigger-http \\
--runtime nodejs8 --entry-point=helloWorld
```

- Use as

```
http://<location>/my-first-function?message=BAM
```

A simple example from https://www.scalyr.com/blog/simple-detailed-introduction-google-cloud-functions/

- Package.json

```
{ "name": "my-first-function", "version": "0.0.1" }
```

- Code

```
exports.helloWorld = (req, res) => {
    let message = req.query.message ||
        req.body.message || 'Hello World!';
    res.status(200).send(message);
};
```

- Deploy with

```
gcloud functions deploy my-first-function --trigger-http \\
--runtime nodejs8 --entry-point=helloWorld
```

- Use as

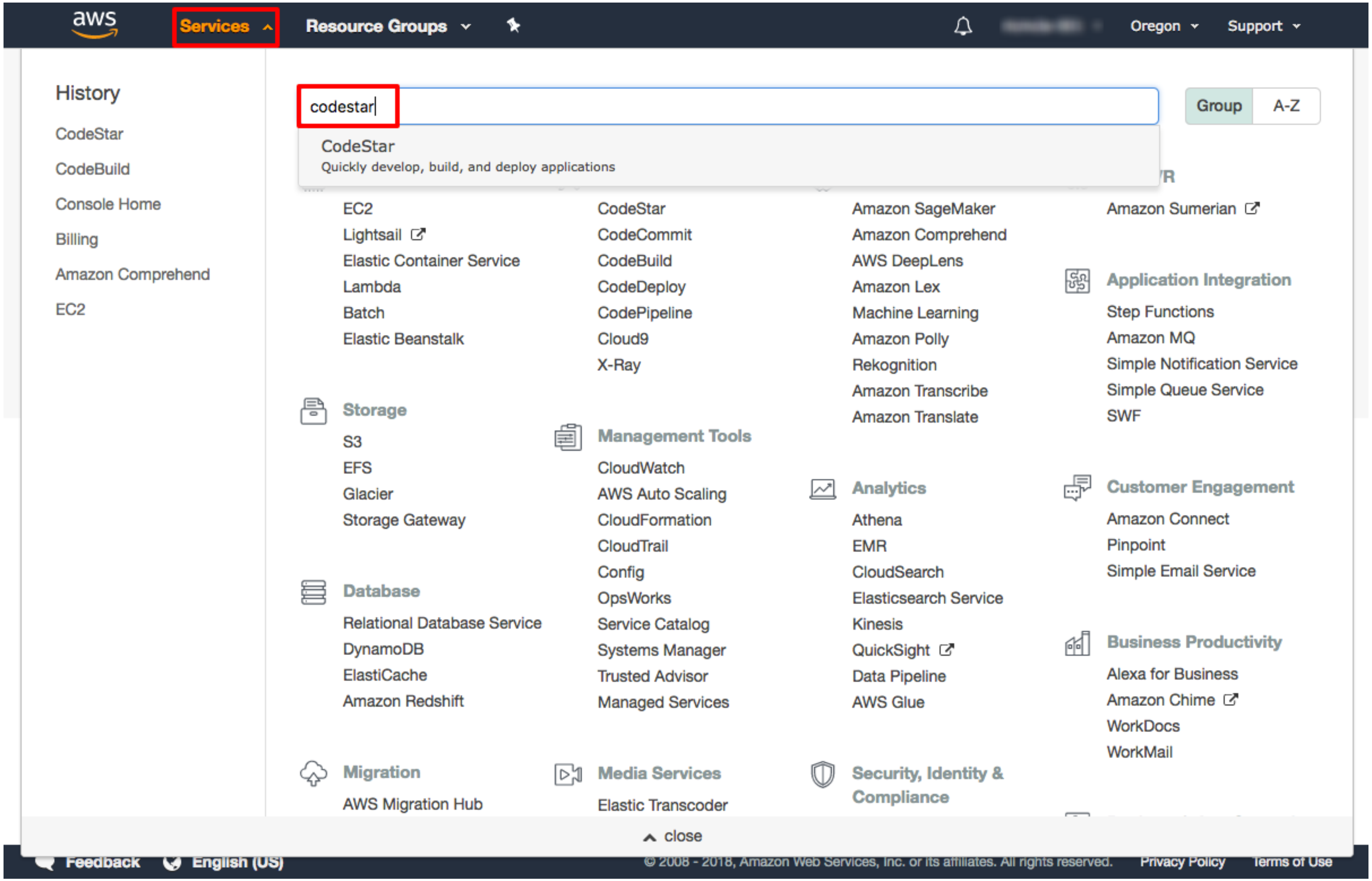
```
http://<location>/my-first-function?message=BAM
```

Something to do with functional programming?

Tampereen yliopisto
Tampere University

The actions with AWS Lambda

https://aws.amazon.com/getting-started/tutorials/build-serverless-app-codestar-cloud9


Services | Resource Groups | [redacted] | Oregon | Support

History
- CodeStar
- CodeBuild
- Console Home
- Billing
- Amazon Comprehend
- EC2

codestar

CodeStar
Quickly develop, build, and deploy applications

Group | A-Z

EC2
Lightsail
Elastic Container Service
Lambda
Batch
Elastic Beanstalk

CodeStar
CodeCommit
CodeBuild
CodeDeploy
CodePipeline
Cloud9
X-Ray

Amazon SageMaker
Amazon Comprehend
AWS DeepLens
Amazon Lex
Machine Learning
Amazon Polly
Rekognition
Amazon Transcribe
Amazon Translate

Amazon Sumerian

Application Integration
Step Functions
Amazon MQ
Simple Notification Service
Simple Queue Service
SWF

Storage
S3
EFS
Glacier
Storage Gateway

Management Tools
CloudWatch
AWS Auto Scaling
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Managed Services

Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
QuickSight
Data Pipeline
AWS Glue

Customer Engagement
Amazon Connect
Pinpoint
Simple Email Service

Database
Relational Database Service
DynamoDB
ElastiCache
Amazon Redshift

Business Productivity
Alexa for Business
Amazon Chime
WorkDocs
WorkMail

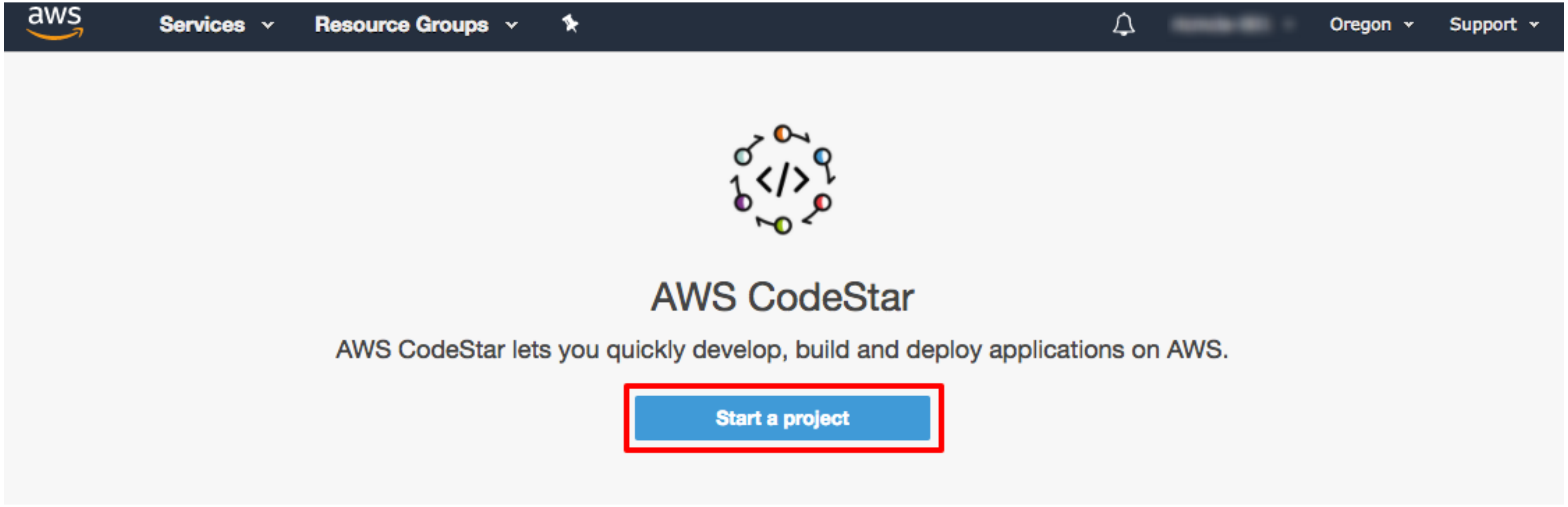
Migration
AWS Migration Hub

Media Services
Elastic Transcoder

Security, Identity & Compliance

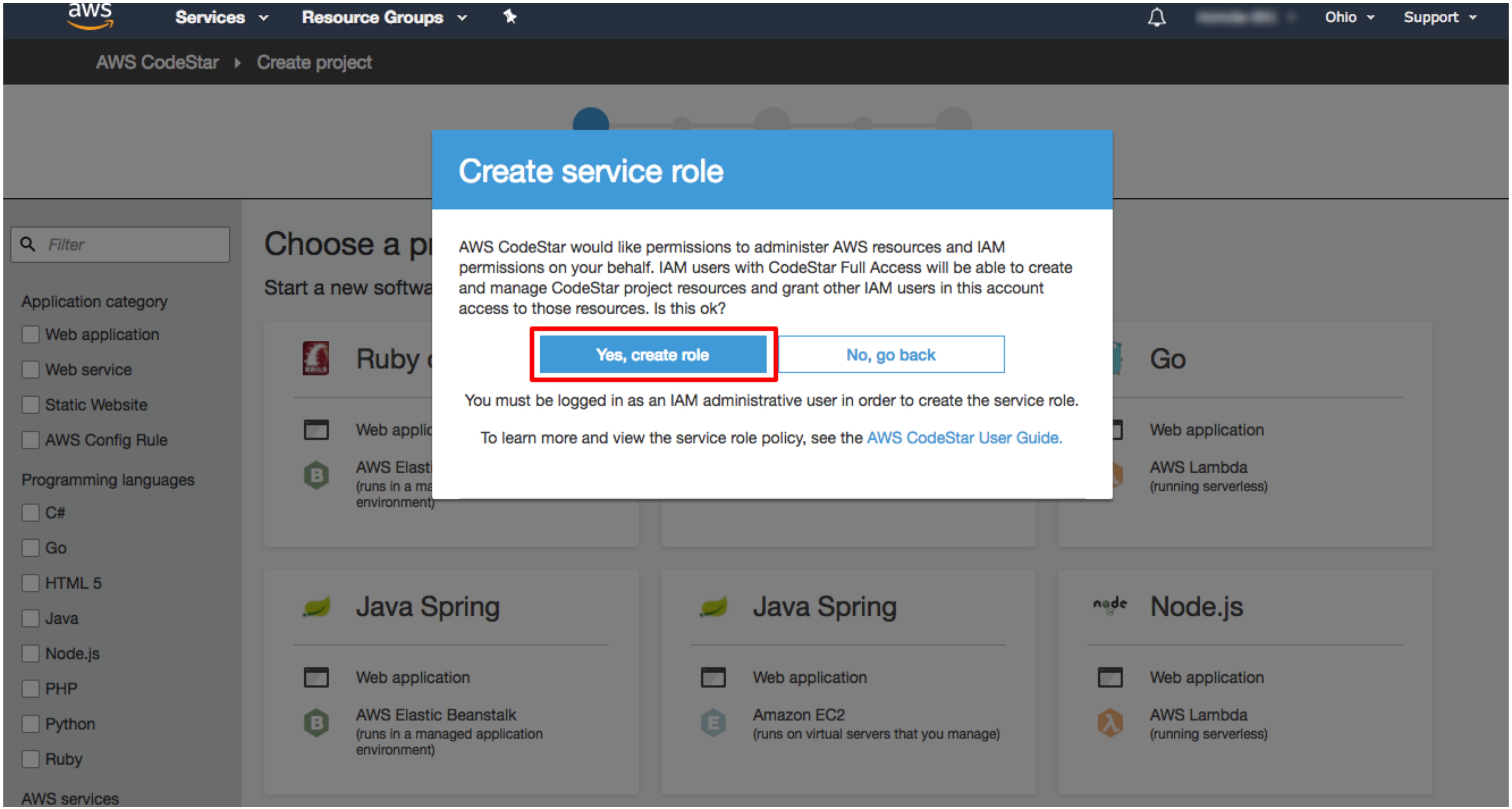
close

Feedback | English (US) | © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use



AWS CodeStar

AWS CodeStar lets you quickly develop, build and deploy applications on AWS.

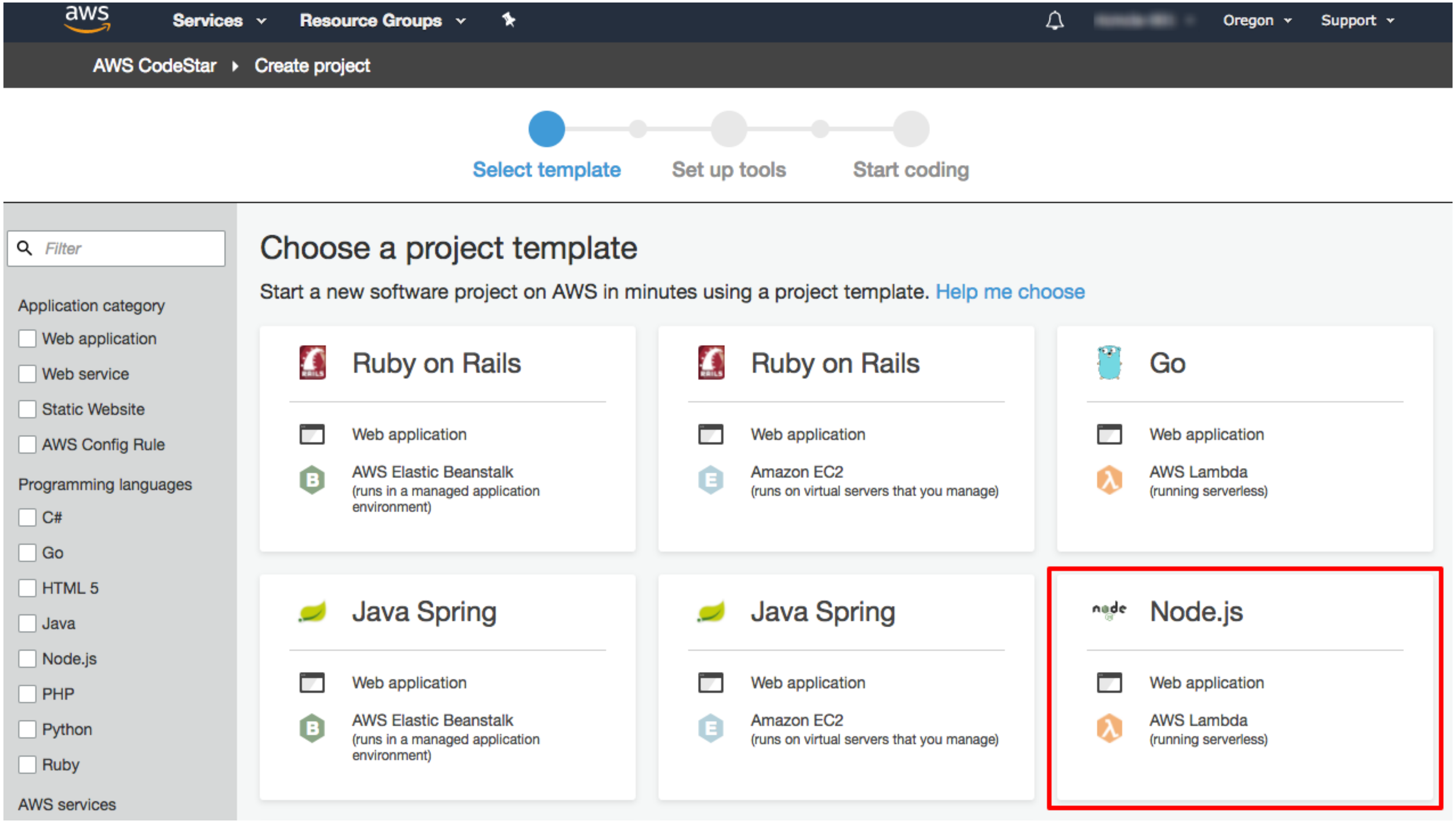


Create service role

AWS CodeStar would like permissions to administer AWS resources and IAM permissions on your behalf. IAM users with CodeStar Full Access will be able to create and manage CodeStar project resources and grant other IAM users in this account access to those resources. Is this ok?

You must be logged in as an IAM administrative user in order to create the service role.

To learn more and view the service role policy, see the AWS CodeStar User Guide.



Choose a project template

Start a new software project on AWS in minutes using a project template. Help me choose

Application category	Programming languages	AWS services
☐ Web application	☐ C#	
☐ Web service	☐ Go	
☐ Static Website	☐ HTML 5	
☐ AWS Config Rule	☐ Java	
	☐ Node.js	
	☐ PHP	
	☐ Python	
	☐ Ruby	

Ruby on Rails
Web application
AWS Elastic Beanstalk
(runs in a managed application environment)

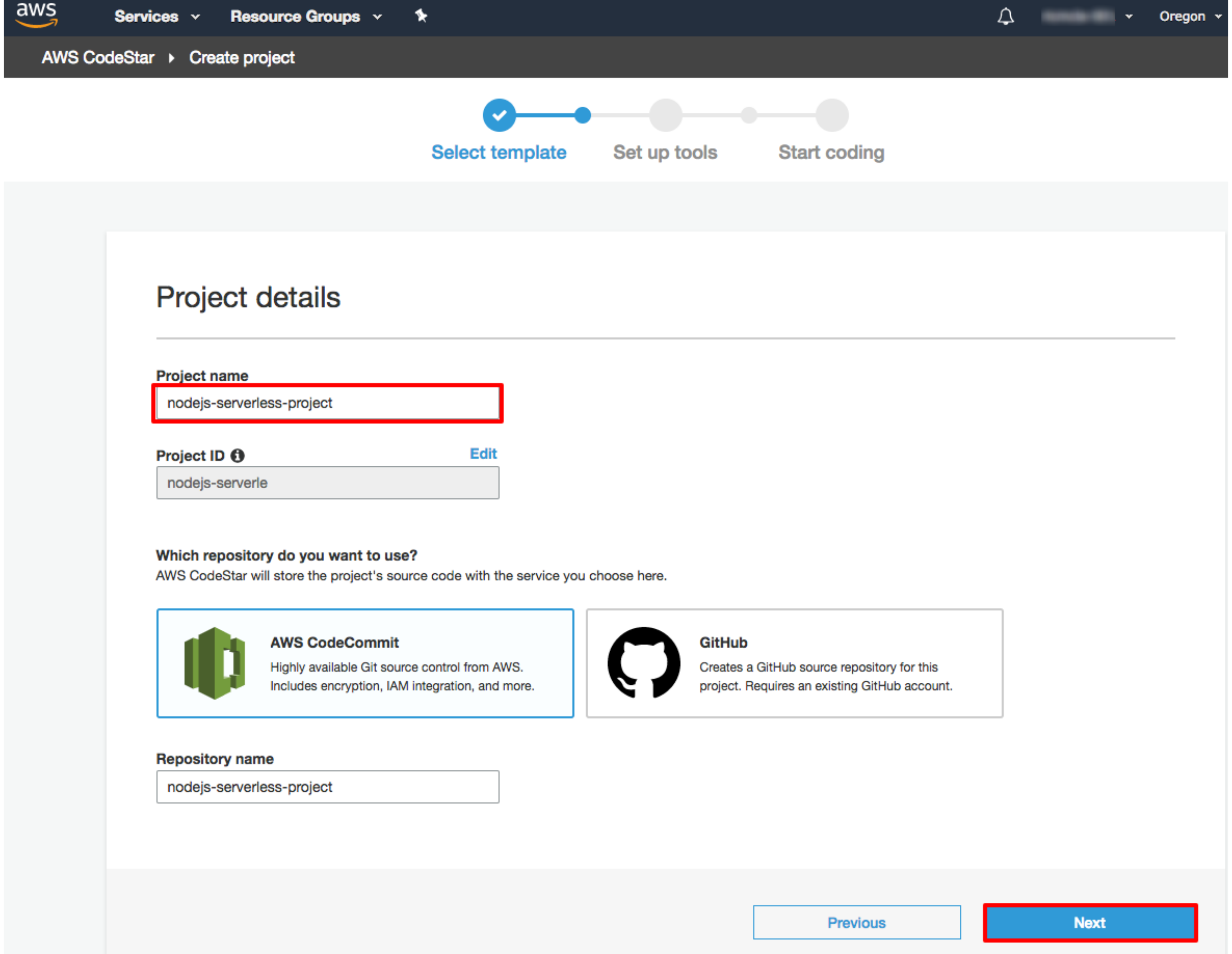
Ruby on Rails
Web application
Amazon EC2
(runs on virtual servers that you manage)

Go
Web application
AWS Lambda
(running serverless)

Java Spring
Web application
AWS Elastic Beanstalk
(runs in a managed application environment)

Java Spring
Web application
Amazon EC2
(runs on virtual servers that you manage)

Node.js
Web application
AWS Lambda
(running serverless)



Project details

Project name: nodejs-serverless-project

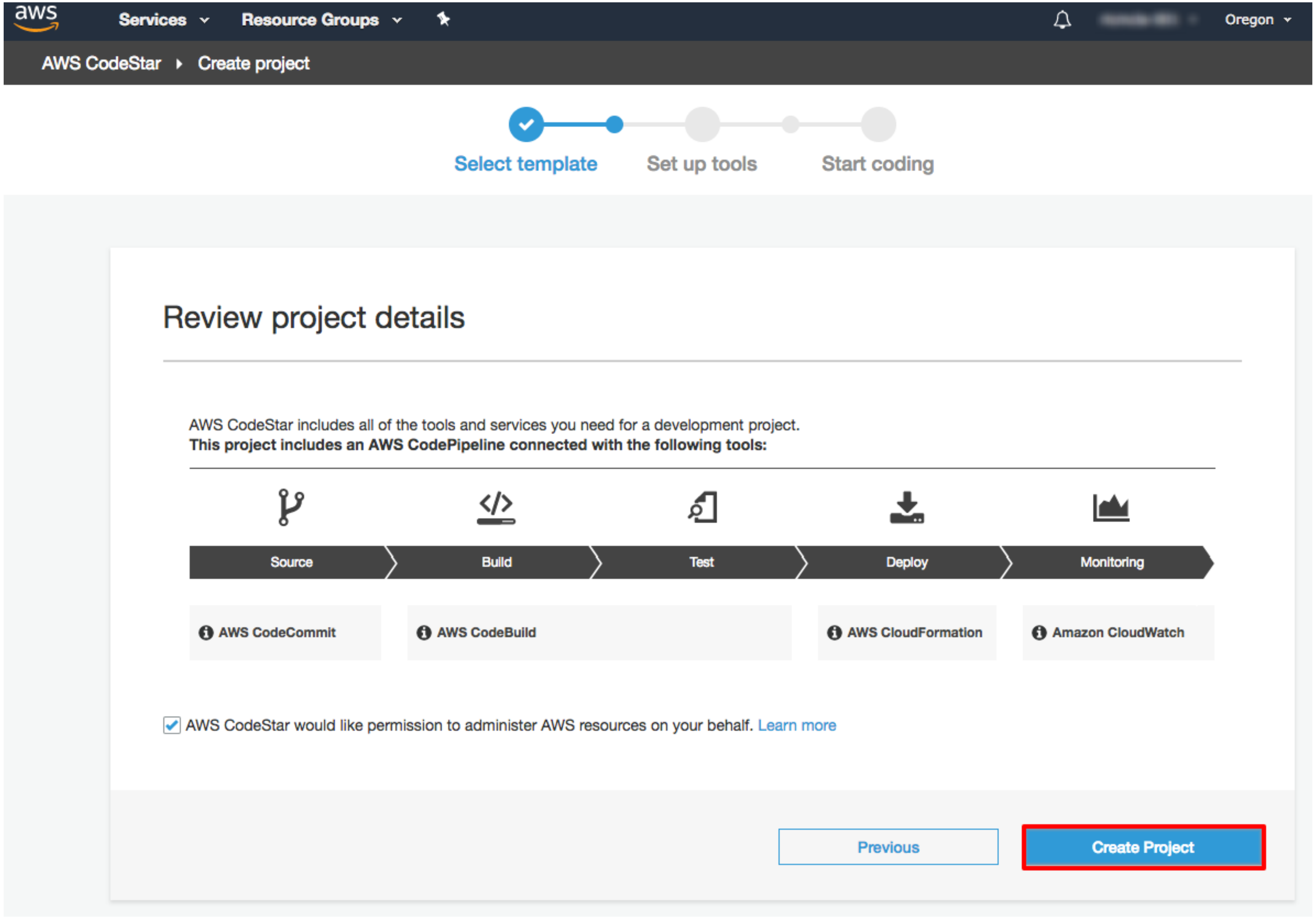
Project ID: nodejs-serverle

Which repository do you want to use?
AWS CodeStar will store the project's source code with the service you choose here.

AWS CodeCommit
Highly available Git source control from AWS. Includes encryption, IAM integration, and more.

GitHub
Creates a GitHub source repository for this project. Requires an existing GitHub account.

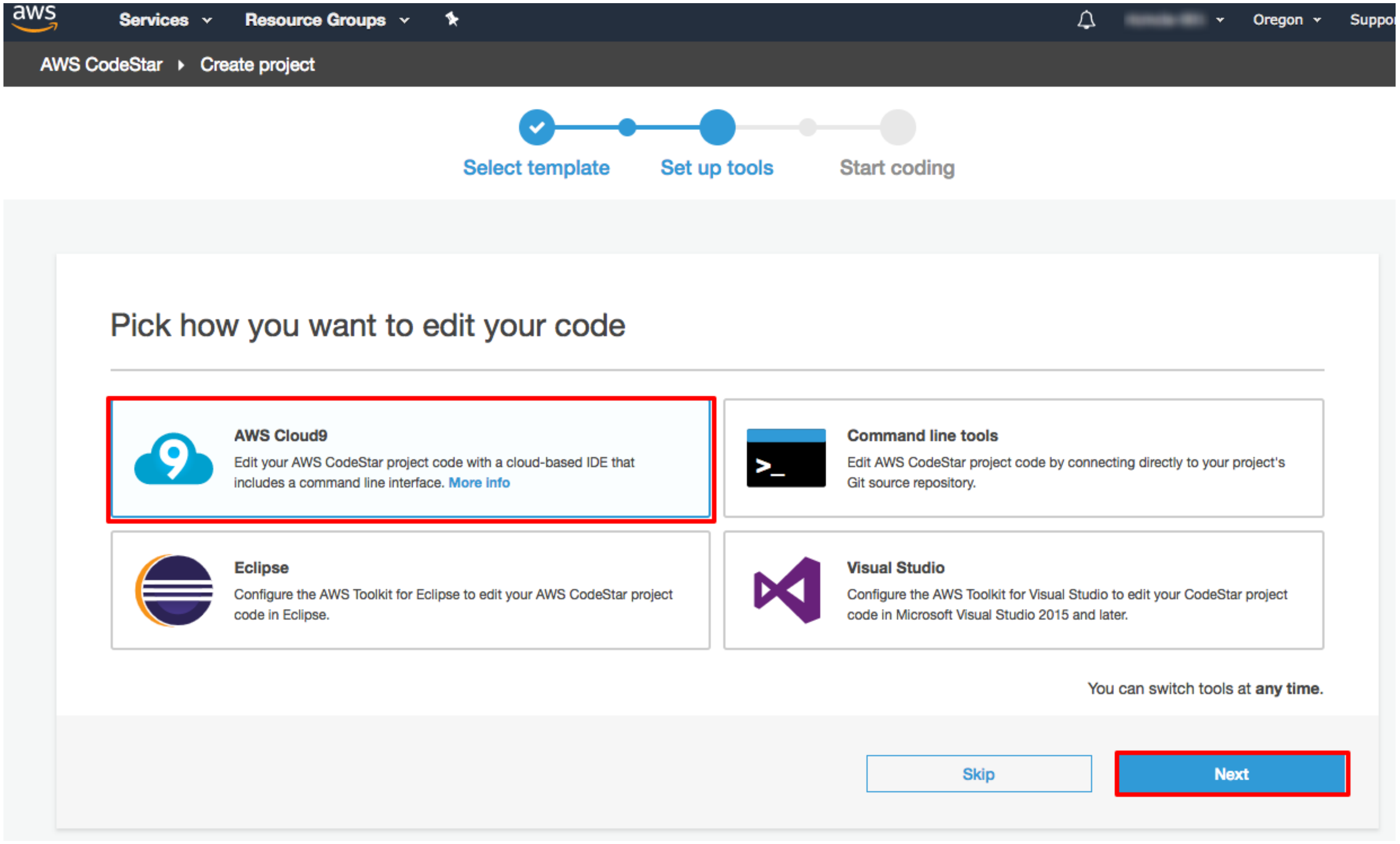
Repository name: nodejs-serverless-project



Review project details

AWS CodeStar includes all of the tools and services you need for a development project.
This project includes an AWS CodePipeline connected with the following tools:

✓ AWS CodeStar would like permission to administer AWS resources on your behalf. Learn more



Pick how you want to edit your code

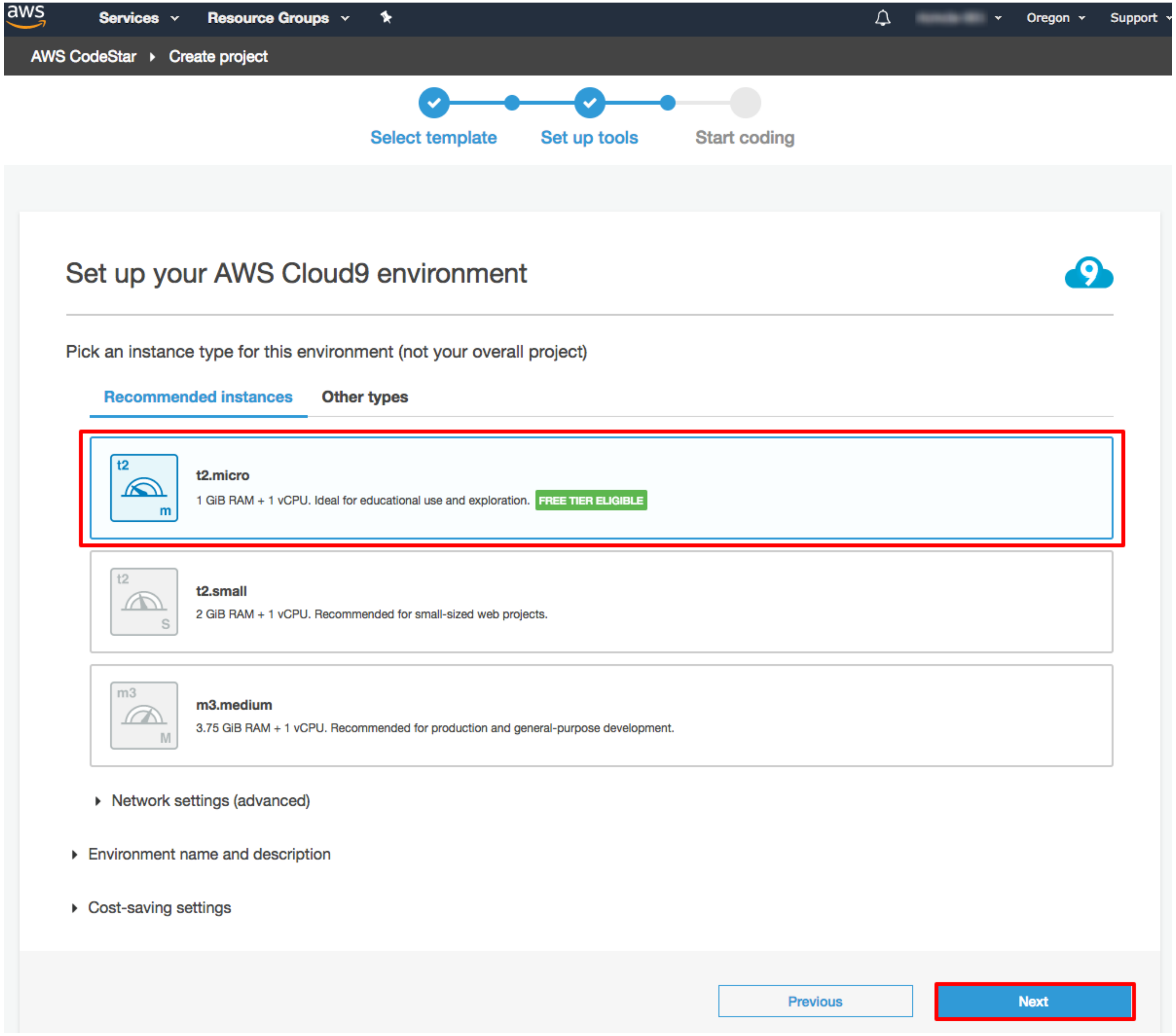
AWS Cloud9
Edit your AWS CodeStar project code with a cloud-based IDE that includes a command line interface. More info

Command line tools
Edit AWS CodeStar project code by connecting directly to your project's Git source repository.

Eclipse
Configure the AWS Toolkit for Eclipse to edit your AWS CodeStar project code in Eclipse.

Visual Studio
Configure the AWS Toolkit for Visual Studio to edit your CodeStar project code in Microsoft Visual Studio 2015 and later.

You can switch tools at **any time**.



Set up your AWS Cloud9 environment

Pick an instance type for this environment (not your overall project)

t2.micro
1 GiB RAM + 1 vCPU. Ideal for educational use and exploration.

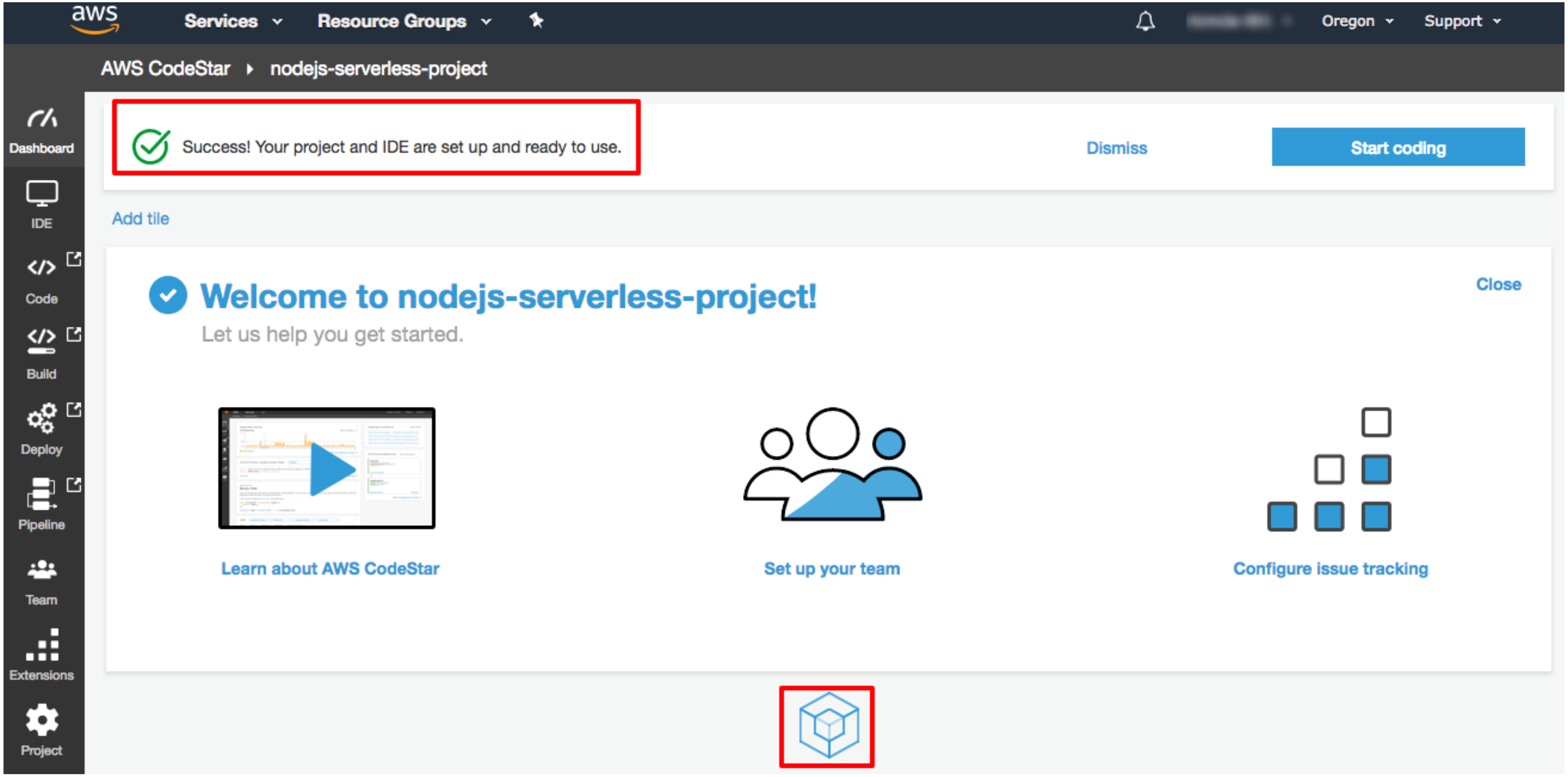
t2.small
2 GiB RAM + 1 vCPU. Recommended for small-sized web projects.

m3.medium
3.75 GiB RAM + 1 vCPU. Recommended for production and general-purpose development.

Network settings (advanced)

Environment name and description

Cost-saving settings



aws Services Resource Groups [redacted] Oregon Support

AWS CodeStar › nodejs-serverless-project

Dashboard
IDE
Code
Build
Deploy
Pipeline
Team
Extensions
Project

Success! Your project and IDE are set up and ready to use.

Dismiss

Start coding

Add tile

Welcome to nodejs-serverless-project!

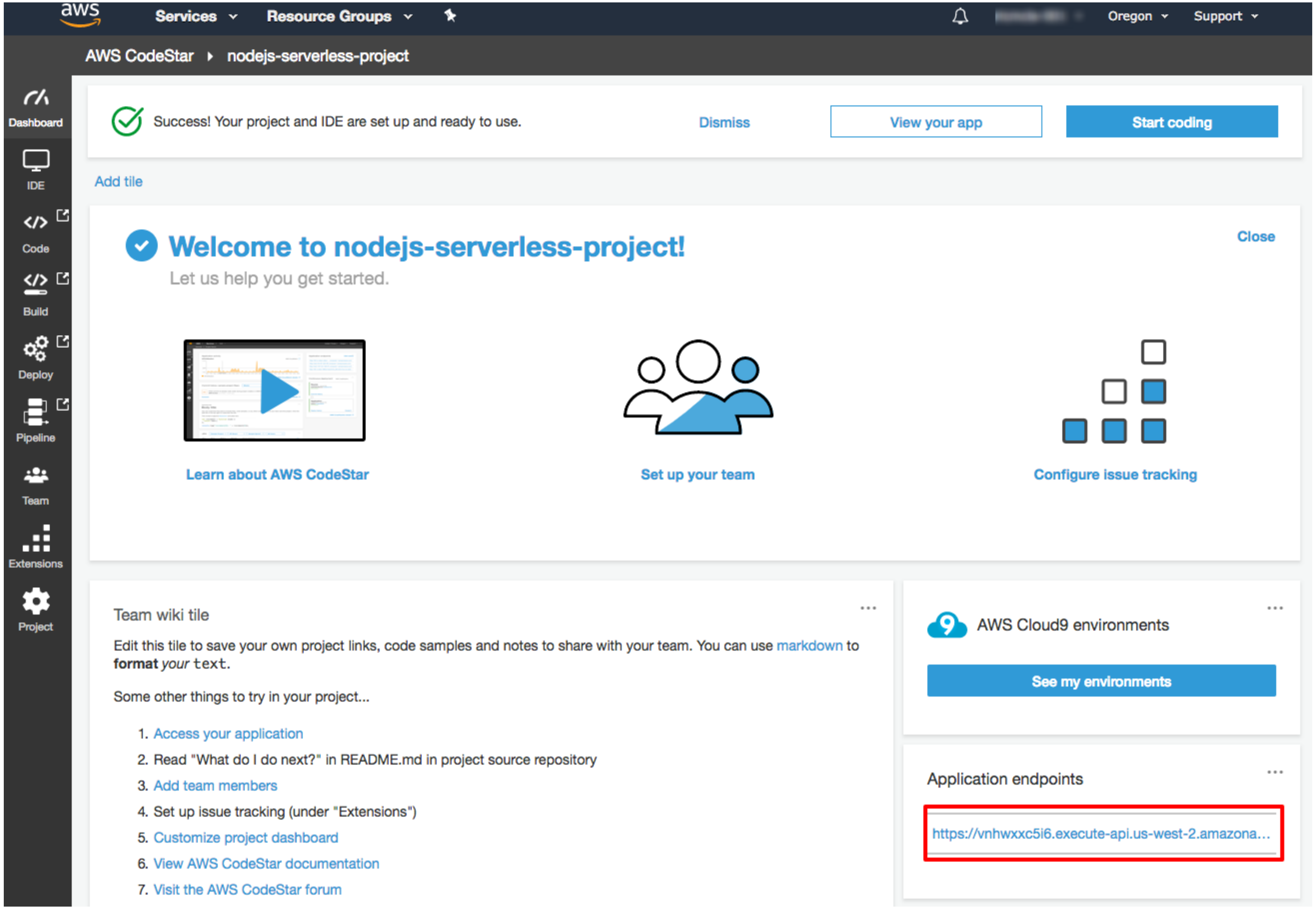
Let us help you get started.

Close

Learn about AWS CodeStar

Set up your team

Configure issue tracking



AWS CodeStar › nodejs-serverless-project

Success! Your project and IDE are set up and ready to use.

Dismiss | View your app | Start coding

Add tile

Welcome to nodejs-serverless-project!

Let us help you get started.

Close

Learn about AWS CodeStar

Set up your team

Configure issue tracking

Team wiki tile

Edit this tile to save your own project links, code samples and notes to share with your team. You can use markdown to **format** *your* `text`.

Some other things to try in your project...

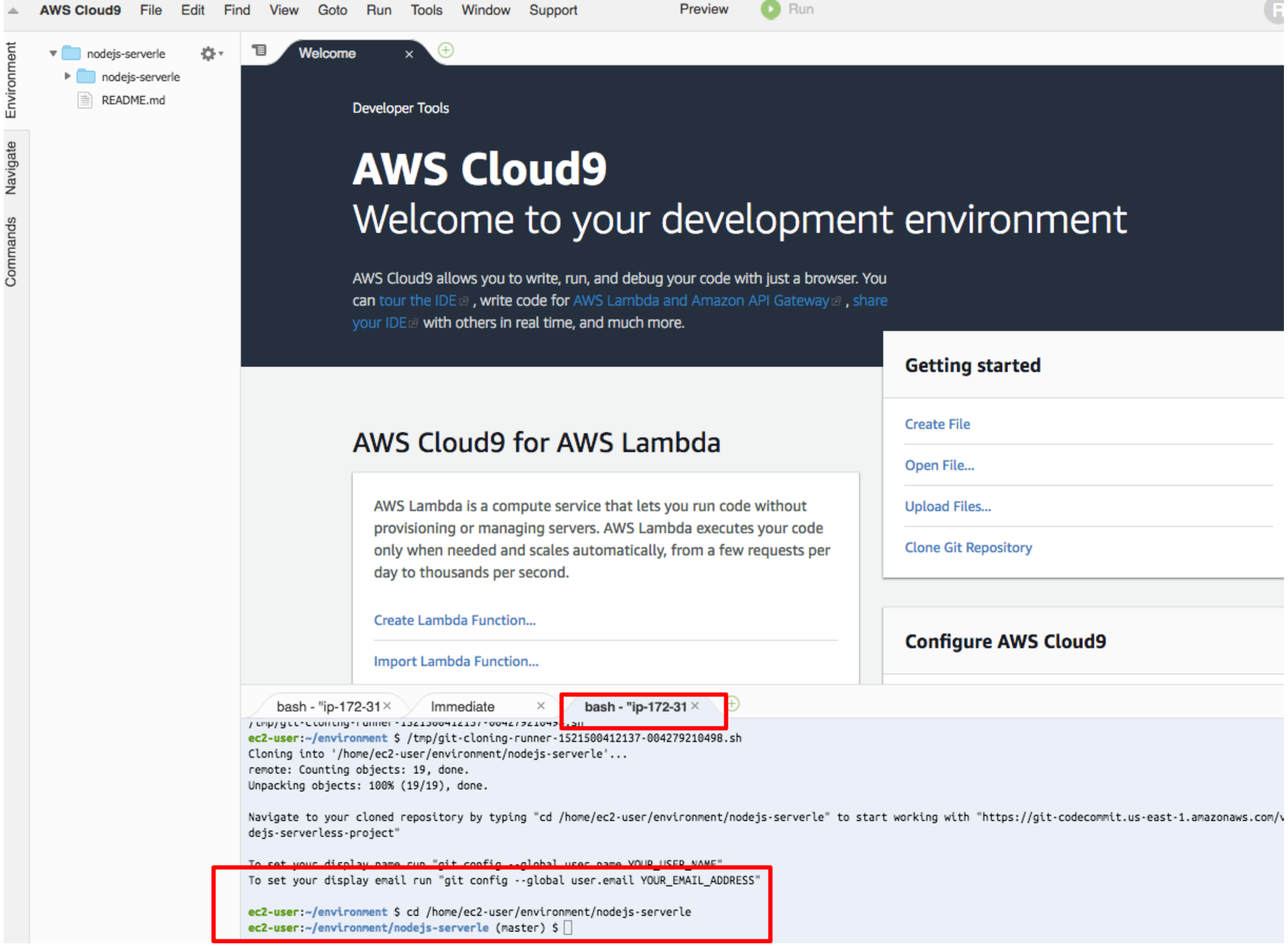
1. Access your application
2. Read "What do I do next?" in README.md in project source repository
3. Add team members
4. Set up issue tracking (under "Extensions")
5. Customize project dashboard
6. View AWS CodeStar documentation
7. Visit the AWS CodeStar forum

AWS Cloud9 environments

See my environments

Application endpoints

https://vnhwxxc5i6.execute-api.us-west-2.amazona...



Developer Tools

AWS Cloud9

Welcome to your development environment

AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can tour the IDE, write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.

Getting started

Create File

Open File...

Upload Files...

Clone Git Repository

AWS Cloud9 for AWS Lambda

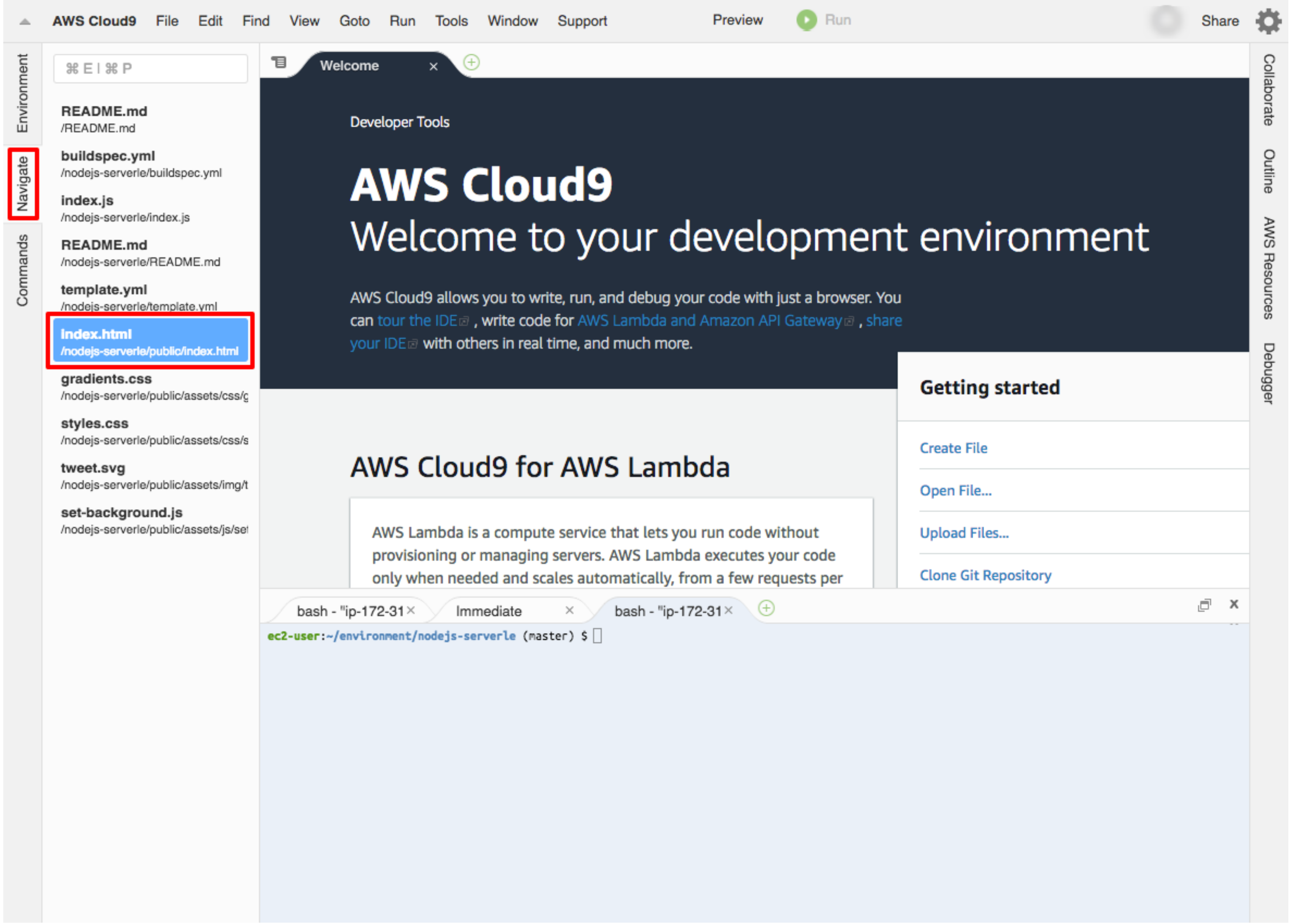
AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second.

Create Lambda Function...

Import Lambda Function...

Configure AWS Cloud9

```
/tmp/git-cloning-runner-1521500412137-004279210498.sh
ec2-user:~/environment $ /tmp/git-cloning-runner-1521500412137-004279210498.sh
Cloning into '/home/ec2-user/environment/nodejs-serverle'...
remote: Counting objects: 19, done.
Unpacking objects: 100% (19/19), done.

Navigate to your cloned repository by typing "cd /home/ec2-user/environment/nodejs-serverle" to start working with "https://git-codecommit.us-east-1.amazonaws.com/v
dejs-serverless-project"

To set your display name run "git config --global user.name YOUR_USER_NAME"
To set your display email run "git config --global user.email YOUR_EMAIL_ADDRESS"

ec2-user:~/environment $ cd /home/ec2-user/environment/nodejs-serverle
ec2-user:~/environment/nodejs-serverle (master) $ 
```



README.md
/README.md
buildspec.yml
/nodejs-serverle/buildspec.yml
index.js
/nodejs-serverle/index.js
README.md
/nodejs-serverle/README.md
template.yml
/nodejs-serverle/template.yml
index.html
/nodejs-serverle/public/index.html
gradients.css
/nodejs-serverle/public/assets/css/g
styles.css
/nodejs-serverle/public/assets/css/s
tweet.svg
/nodejs-serverle/public/assets/img/t
set-background.js
/nodejs-serverle/public/assets/js/set

Developer Tools

AWS Cloud9
Welcome to your development environment

AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can tour the IDE, write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.

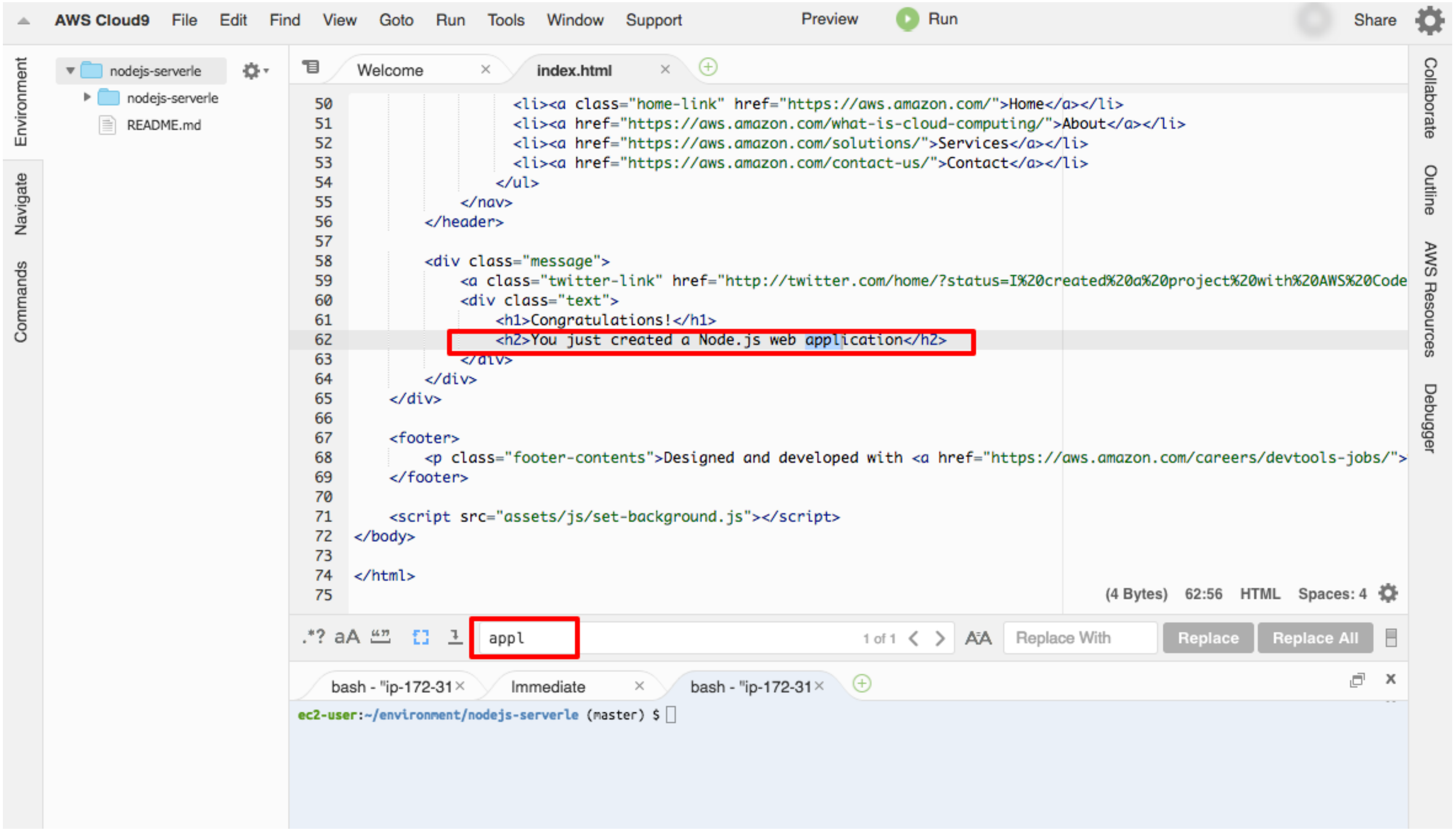
Getting started

Create File
Open File...
Upload Files...
Clone Git Repository

AWS Cloud9 for AWS Lambda

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code only when needed and scales automatically, from a few requests per

```
ec2-user:~/environment/nodejs-serverle (master) $
```

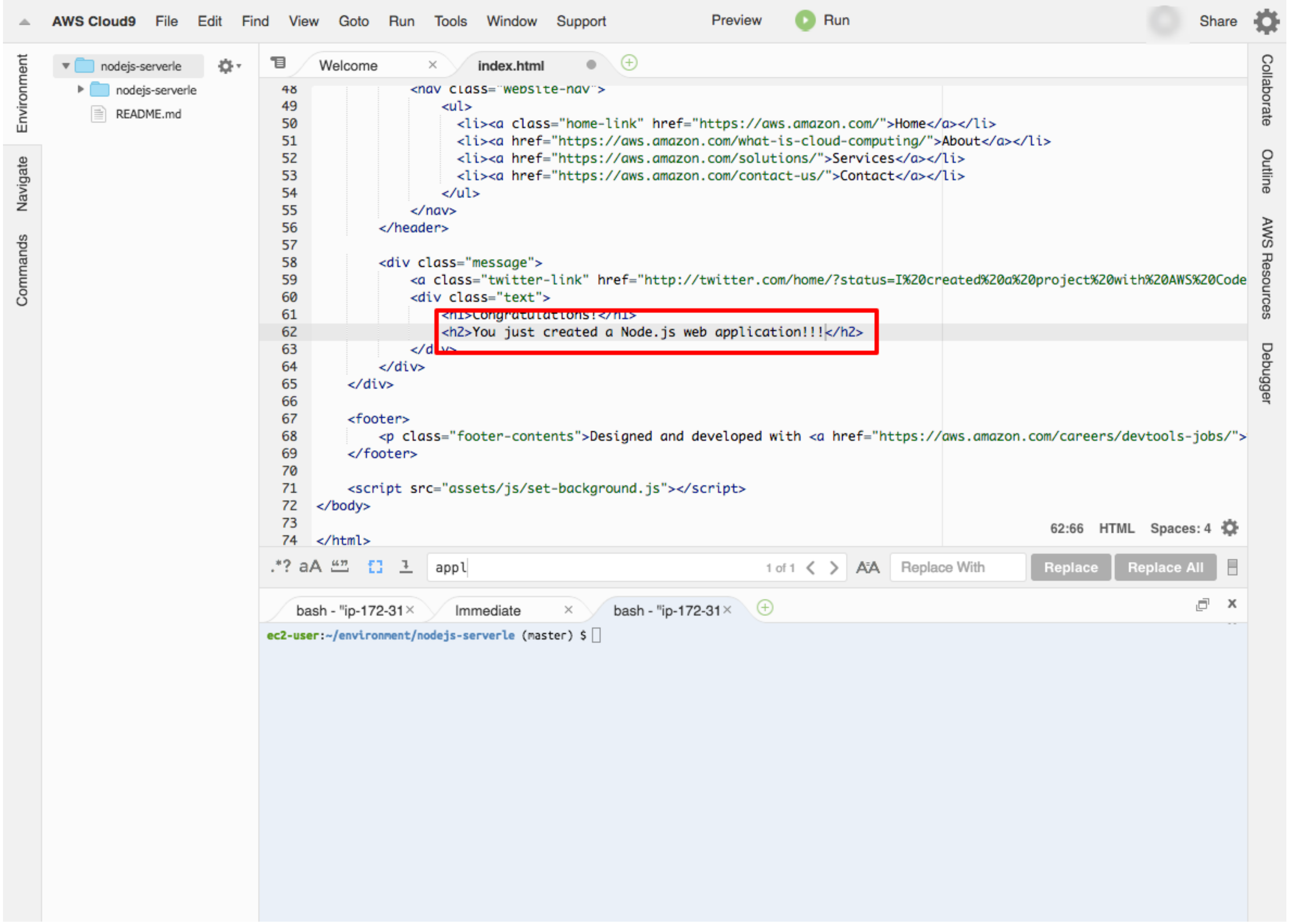


```
                <li><a class="home-link" href="https://aws.amazon.com/">Home</a></li>
                <li><a href="https://aws.amazon.com/what-is-cloud-computing/">About</a></li>
                <li><a href="https://aws.amazon.com/solutions/">Services</a></li>
                <li><a href="https://aws.amazon.com/contact-us/">Contact</a></li>
                </ul>
            </nav>
        </header>

        <div class="message">
            <a class="twitter-link" href="http://twitter.com/home/?status=I%20created%20a%20project%20with%20AWS%20Code
            <div class="text">
                <h1>Congratulations!</h1>
                <h2>You just created a Node.js web application</h2>
            </div>
        </div>
    </div>

    <footer>
        <p class="footer-contents">Designed and developed with <a href="https://aws.amazon.com/careers/devtools-jobs/">
    </footer>

    <script src="assets/js/set-background.js"></script>
</body>

</html>
```

(4 Bytes) 62:56 HTML Spaces: 4

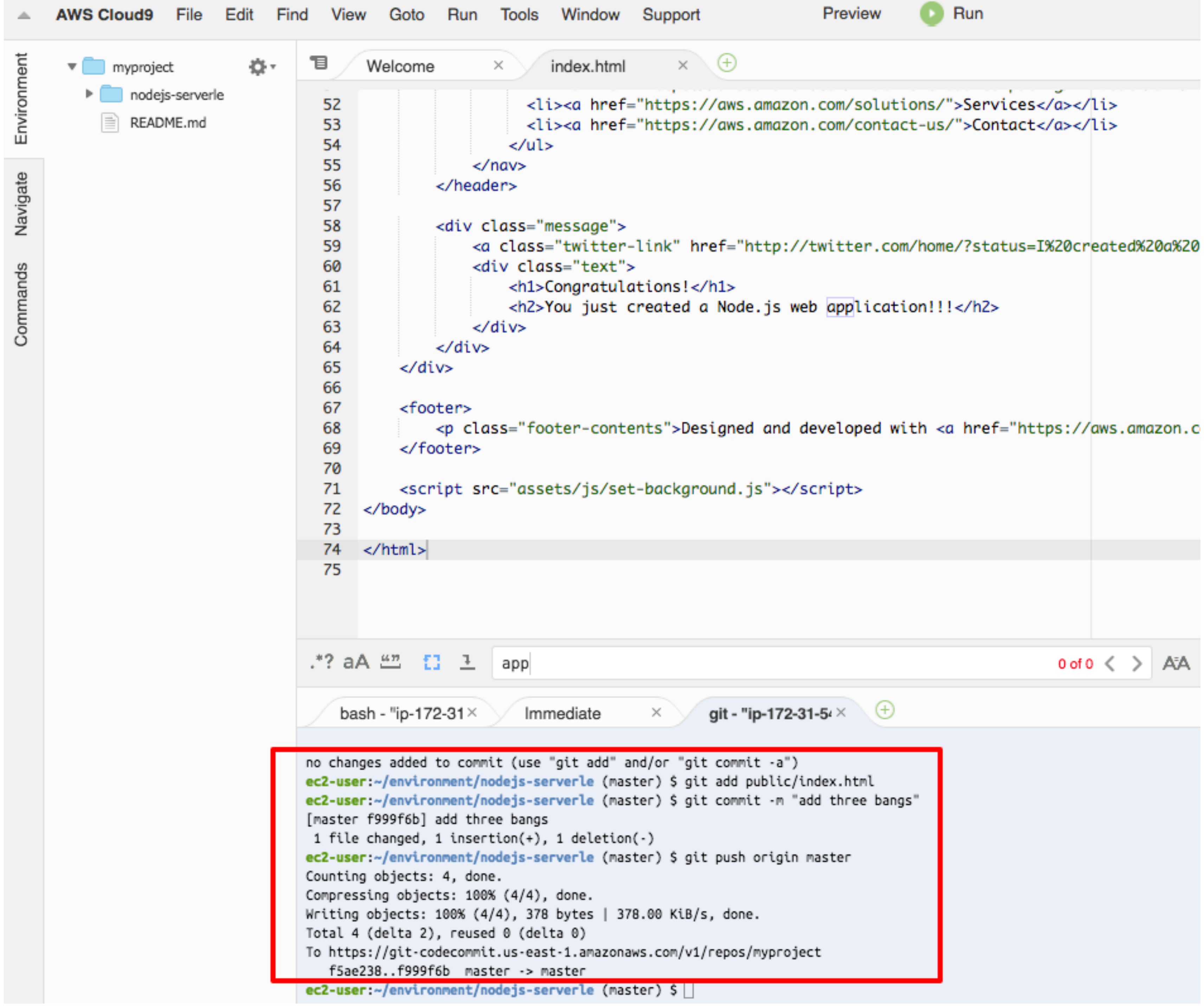
appl 1 of 1

```
ec2-user:~/environment/nodejs-serverle (master) $
```



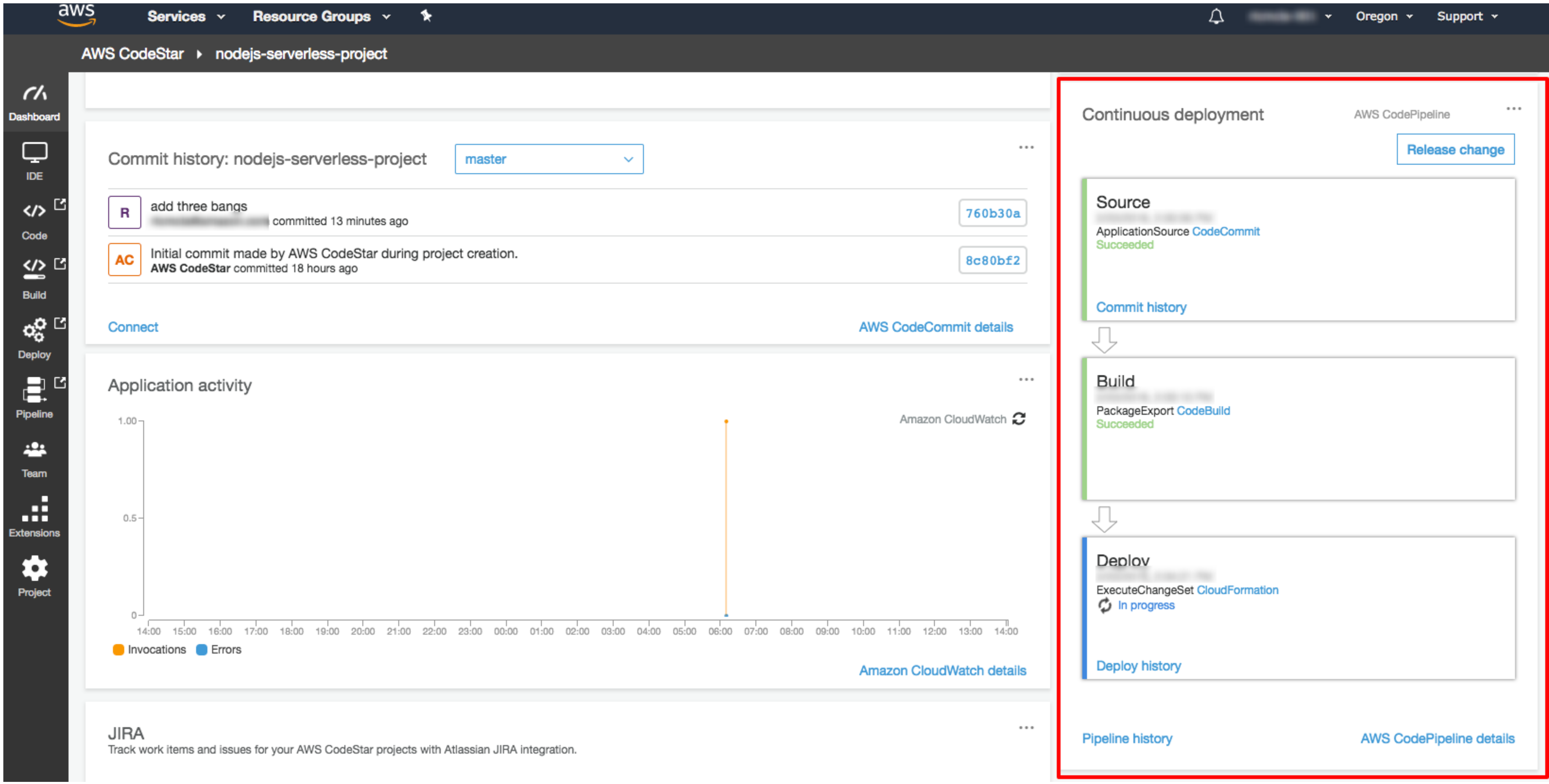
```
            <nav class="website-nav">
                <ul>
                  <li><a class="home-link" href="https://aws.amazon.com/">Home</a></li>
                  <li><a href="https://aws.amazon.com/what-is-cloud-computing/">About</a></li>
                  <li><a href="https://aws.amazon.com/solutions/">Services</a></li>
                  <li><a href="https://aws.amazon.com/contact-us/">Contact</a></li>
                </ul>
            </nav>
        </header>

        <div class="message">
            <a class="twitter-link" href="http://twitter.com/home/?status=I%20created%20a%20project%20with%20AWS%20Code
            <div class="text">
                <h1>Congratulations!</h1>
                <h2>You just created a Node.js web application!!!</h2>
            </div>
        </div>
    </div>

    <footer>
        <p class="footer-contents">Designed and developed with <a href="https://aws.amazon.com/careers/devtools-jobs/">
    </footer>

    <script src="assets/js/set-background.js"></script>
</body>

</html>
```

```
ec2-user:~/environment/nodejs-serverle (master) $
```



```
                    <li><a href="https://aws.amazon.com/solutions/">Services</a></li>
                    <li><a href="https://aws.amazon.com/contact-us/">Contact</a></li>
                </ul>
            </nav>
        </header>

        <div class="message">
            <a class="twitter-link" href="http://twitter.com/home/?status=I%20created%20a%20
            <div class="text">
                <h1>Congratulations!</h1>
                <h2>You just created a Node.js web application!!!</h2>
            </div>
        </div>
    </div>

    <footer>
        <p class="footer-contents">Designed and developed with <a href="https://aws.amazon.c
    </footer>

    <script src="assets/js/set-background.js"></script>
</body>

</html>
```

```
no changes added to commit (use "git add" and/or "git commit -a")
ec2-user:~/environment/nodejs-serverle (master) $ git add public/index.html
ec2-user:~/environment/nodejs-serverle (master) $ git commit -m "add three bangs"
[master f999f6b] add three bangs
 1 file changed, 1 insertion(+), 1 deletion(-)
ec2-user:~/environment/nodejs-serverle (master) $ git push origin master
Counting objects: 4, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 378 bytes | 378.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0)
To https://git-codecommit.us-east-1.amazonaws.com/v1/repos/myproject
   f5ae238..f999f6b  master -> master
ec2-user:~/environment/nodejs-serverle (master) $
```



AWS CodeStar › nodejs-serverless-project

Commit history: nodejs-serverless-project   master

R	add three bangs [redacted] committed 13 minutes ago	760b30a
AC	Initial commit made by AWS CodeStar during project creation. **AWS CodeStar** committed 18 hours ago	8c80bf2

Connect   AWS CodeCommit details

Application activity   Amazon CloudWatch

Amazon CloudWatch details

JIRA
Track work items and issues for your AWS CodeStar projects with Atlassian JIRA integration.

Continuous deployment   AWS CodePipeline

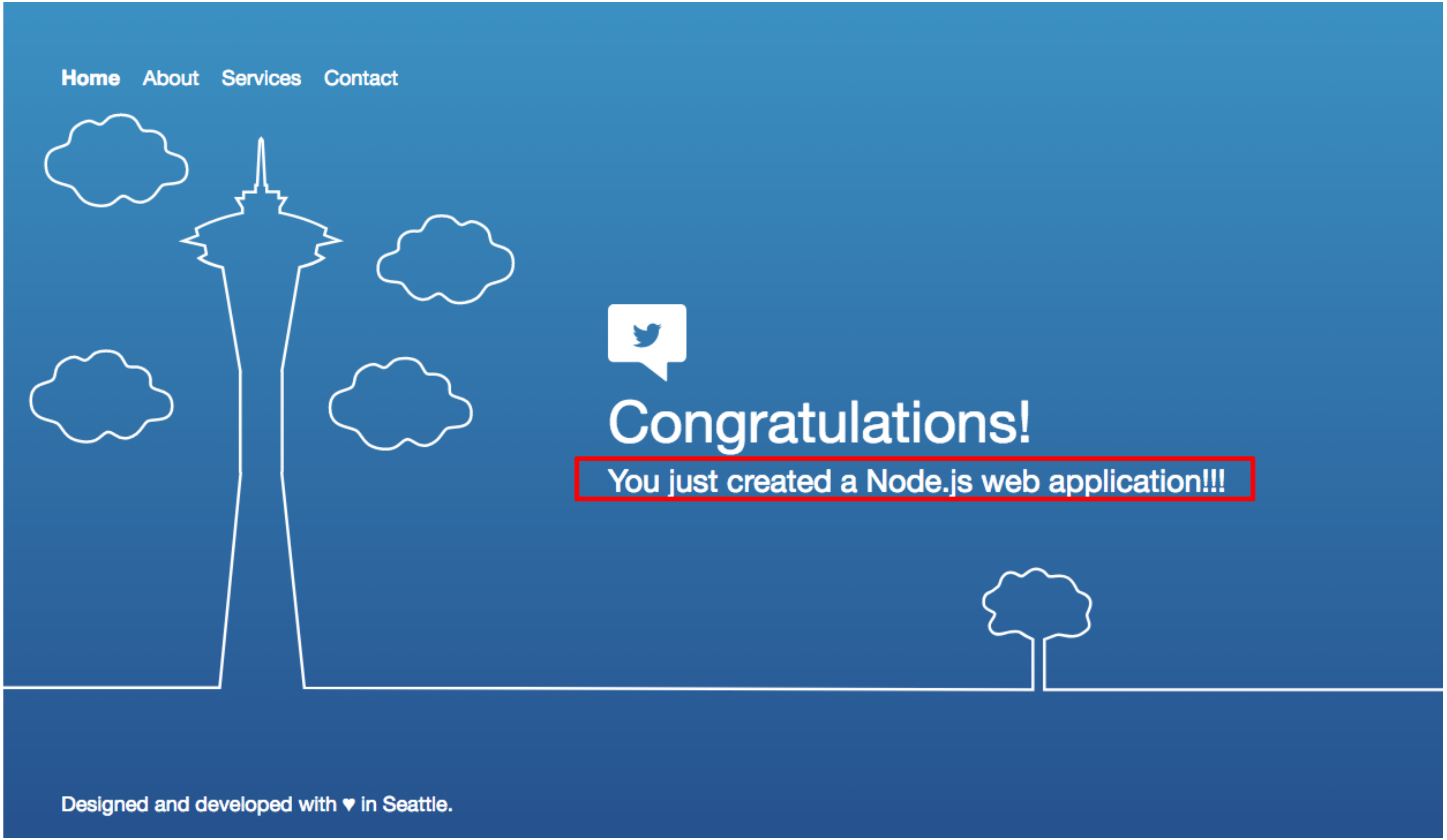
Release change

Source
[redacted]
ApplicationSource CodeCommit
Succeeded
Commit history

Build
[redacted]
PackageExport CodeBuild
Succeeded

Deploy
[redacted]
ExecuteChangeSet CloudFormation
In progress
Deploy history

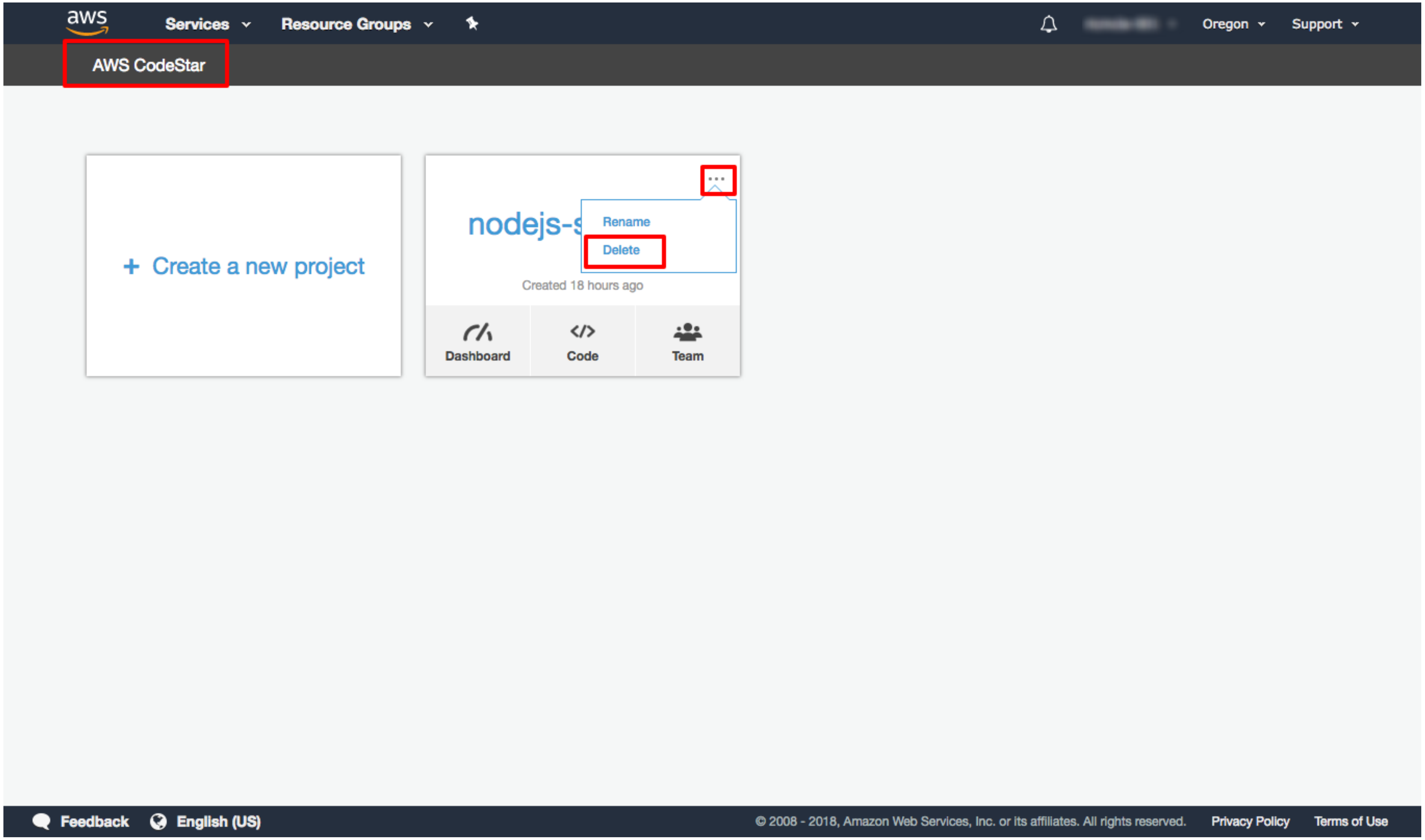
Pipeline history   AWS CodePipeline details



Congratulations!

You just created a Node.js web application!!!

Designed and developed with ♥ in Seattle.


aws
Services
Resource Groups
Oregon
Support
AWS CodeStar
+ Create a new project
nodejs-s
Rename
Delete
Created 18 hours ago
Dashboard
Code
Team
Feedback
English (US)
© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Claimed FaaS advantages

- Smaller for developer since infrastructure is handled by somebody else
  => more time for writing application code
- Inherently scalable
- No need to pay for idle resources
  (temptation to miss-use)
- Available and fault tolerant
- No explicit multi-tenancy
- Forces modular business logic

Claimed FaaS disadvantages

- Decreased transparency
- Maybe challenging to debug
- Autoscaling of functions may lead to autoscaling of cost
- Keeping track of huge numbers of functions is tough
- Chaching of requests?

Nice video about microservices

- Netflix story (Mastering Chaos - A Netflix Guide to Microservices) <https://www.youtube.com/watch?v=CZ3wIuvmHeM>