

CI tools



Jenkins



Travis



codeship



TeamCity



AppVeyor



Circle CI



LambCI



AWS CodePipeline



Concourse CI

design goals

design goals

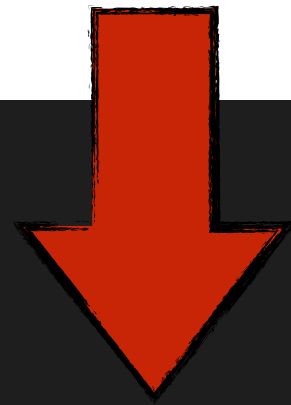
- “pipeline as code”

design goals

- “pipeline as code”
- reproducible builds

reproducible builds

NPM default - get latest
“compatible” version, ie. 1.X.X

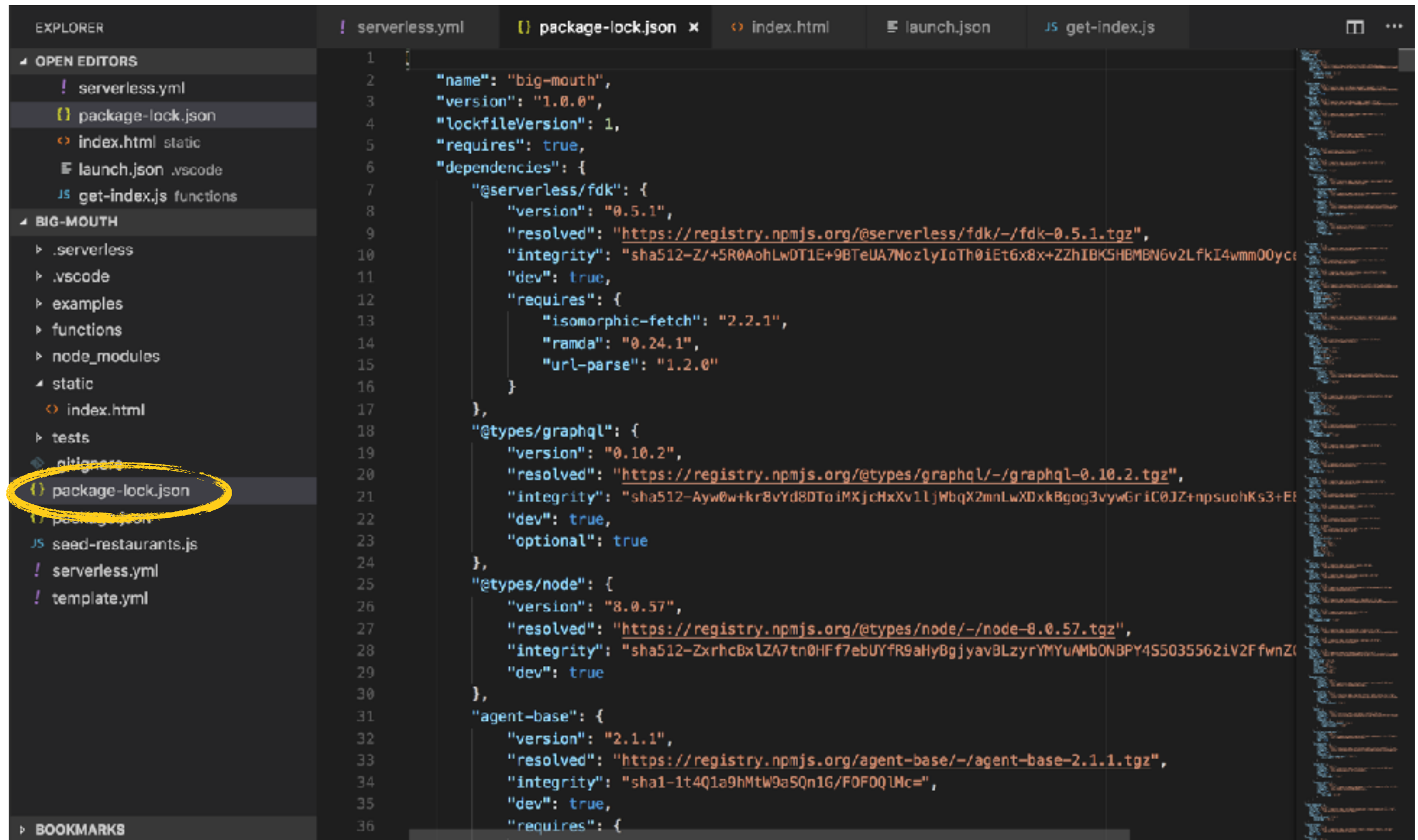


```
license: "ISC",  
"dependencies": {  
  "bluebird": "^3.5.0",  
  "co": "^4.6.0",  
  "do-not-download-this-package": "^1.0.0",  
  "do-not-download-this-package-neither": "^1.0.0"  
},
```

reproducible builds

use NPM shrinkwrap
or, upgrade to NPM v5

reproducible builds



The screenshot shows the VS Code interface with the Explorer, Explorer, and Explorer panels. The Explorer panel on the left shows the project structure, with the `package-lock.json` file highlighted. The Explorer panel in the middle shows the content of `package-lock.json`, which is a JSON file defining the project's dependencies and their resolved versions. The Explorer panel on the right shows the content of `serverless.yml`, which is a YAML file defining the project's serverless configuration. The `package-lock.json` file is highlighted in the Explorer panel, and its content is displayed in the Explorer panel. The `serverless.yml` file is also highlighted in the Explorer panel, and its content is displayed in the Explorer panel. The `package-lock.json` file is a JSON file that defines the project's dependencies and their resolved versions. It includes the project name, version, lockfile version, and a list of dependencies with their resolved versions and integrity hashes. The `serverless.yml` file is a YAML file that defines the project's serverless configuration, including the name, version, and dependencies.

```
1  {
2    "name": "big-mouth",
3    "version": "1.0.0",
4    "lockfileVersion": 1,
5    "requires": true,
6    "dependencies": {
7      "@serverless/fdk": {
8        "version": "0.5.1",
9        "resolved": "https://registry.npmjs.org/@serverless/fdk/-/fdk-0.5.1.tgz",
10       "integrity": "sha512-Z/+5R0AohLwDT1E+9BTeUA7NozlyIoTh0iEt6x8x+ZZhIBK5HBMBN6v2LfkI4wmm00yc",
11       "dev": true,
12       "requires": {
13         "isomorphic-fetch": "2.2.1",
14         "ramda": "0.24.1",
15         "url-parse": "1.2.0"
16       }
17     },
18     "@types/graphql": {
19       "version": "0.10.2",
20       "resolved": "https://registry.npmjs.org/@types/graphql/-/graphql-0.10.2.tgz",
21       "integrity": "sha512-Ayw0w+kr8vYd8DT0iMXjcHxXv1ljWbqX2mnLwXDxkBgog3vywGr iC0JZ+npsuohKs3+EF",
22       "dev": true,
23       "optional": true
24     },
25     "@types/node": {
26       "version": "8.0.57",
27       "resolved": "https://registry.npmjs.org/@types/node/-/node-8.0.57.tgz",
28       "integrity": "sha512-ZxrhcBx1ZA7tn0HFf7ebUYfr9aHyBgjyavBLzyrYMYuAMBONBPY4S5035562iV2FfwnZ",
29       "dev": true
30     },
31     "agent-base": {
32       "version": "2.1.1",
33       "resolved": "https://registry.npmjs.org/agent-base/-/agent-base-2.1.1.tgz",
34       "integrity": "sha1-1t4Q1a9hMtW9a5Qn1G/F0F0Q1Mc=",
35       "dev": true,
36       "requires": {
```

Postmortem for Malicious Packages Published on July 12th, 2018

Summary

On July 12th, 2018, an attacker compromised the npm account of an ESLint maintainer and published malicious versions of the `eslint-scope` and `eslint-config-eslint` packages to the npm registry. On installation, the malicious packages downloaded and executed code from `pastebin.com` which sent the contents of the user's `.npmrc` file to the attacker. An `.npmrc` file typically contains access tokens for publishing to npm.

The malicious package versions are `eslint-scope@3.7.2` and `eslint-config-eslint@5.0.2`, both of which have been unpublished from npm. The `pastebin.com` paste linked in these packages has also been taken down.

npm has revoked all access tokens issued before 2018-07-12 12:30 UTC. As a result, all access tokens compromised by this attack should no longer be usable.

The maintainer whose account was compromised had reused their npm password on several other sites and did not have two-factor authentication enabled on their npm account.

We, the ESLint team, are sorry for allowing this to happen. We hope that other package maintainers can learn from our mistakes and improve the security of the whole npm ecosystem.

<https://eslint.org/blog/2018/07/postmortem-for-malicious-package-publishes>



xnɔɹɐf uɐɹɔq
@brianleroux



Following



🤔 A few things we npm package authors/maintainers can do today:

- 🔒 Enable 2fa for everyone like rn
- 🔑 Revoke/reset all your tokens *
- 📌 Pin all deps to an exact version in package.json as a practice so auto-upgrades cannot burn us

* Don't forget your CI! 🙏

6:00 PM - 12 Jul 2018

54 Retweets 108 Likes



7



54



108



design goals

- “pipeline as code”
- reproducible builds
- fast!

fast pipeline

- no more being stuck in a queue
- building & testing should be fast

design goals

- “pipeline as code”
- reproducible builds
- fast!
- serverless ;-)

build script

```
if [ "$1" = "deploy" ] && [ $# -eq 4 ]; then
    STAGE=$2
    REGION=$3
    PROFILE=$4

    npm install
    AWS_PROFILE=$PROFILE 'node_modules/.bin/sls' deploy -s $STAGE -r $REGION
elif [ "$1" = "int-test" ] && [ $# -eq 4 ]; then
    STAGE=$2
    REGION=$3
    PROFILE=$4

    npm install
    AWS_PROFILE=$PROFILE npm run int-$STAGE
elif [ "$1" = "acceptance-test" ] && [ $# -eq 4 ]; then
    STAGE=$2
    REGION=$3
    PROFILE=$4

    npm install
    AWS_PROFILE=$PROFILE npm run acceptance-$STAGE
else
    usage
    exit 1
fi
```

build script

Build

Execute shell

Command `./build.sh test test eu-west-1 non_prod`

See [the list of available environment variables](#)

Execute shell

Command `./build.sh deploy test eu-west-1 non_prod`

See [the list of available environment variables](#)

Execute shell

Command `./build.sh validate test eu-west-1 non_prod`

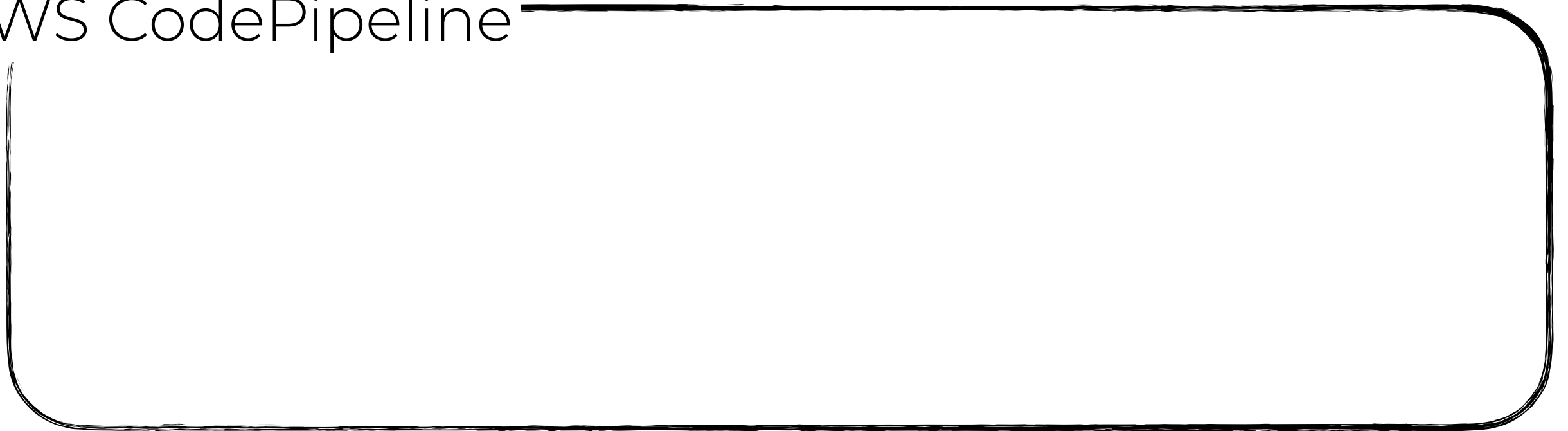
See [the list of available environment variables](#)

CodePipeline demo

recap



AWS CodePipeline



recap



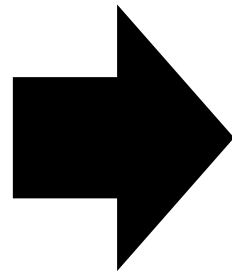
AWS CodePipeline



recap



AWS CodePipeline

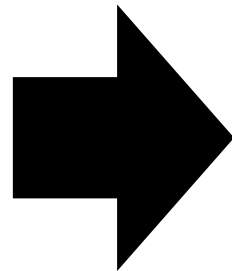


AWS CodeBuild

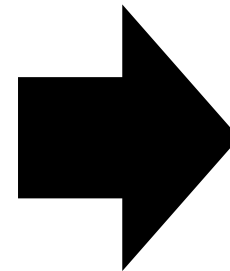
recap



AWS CodePipeline



AWS CodeBuild



integration test

deploy **SERVERLESS**

acceptance test

design goals

- ☐ “pipeline as code”
- ☐ reproducible builds
- ☐ fast!
- ☐ serverless

design goals



“pipeline as code”



reproducible builds



fast!



serverless

design goals



“pipeline as code”



reproducible builds



fast!



serverless

design goals



“pipeline as code”



reproducible builds



fast!



serverless

design goals



“pipeline as code”



reproducible builds



fast!



serverless

serverless pipeline



AWS CodeBuild

Pricing Details

Build Duration

Build Duration is calculated in minutes, from the time you submit your build until your build is terminated, rounded up to the nearest minute.

Compute Types

AWS CodeBuild offers three compute instance types with different amounts of memory and CPU. Charges vary by the compute instance type that you choose for your build.

Compute instance type	Memory (GB)	vCPU	Price per build minute (\$)
build.general1.small	3	2	0.005
build.general1.medium	7	4	0.010
build.general1.large	15	8	0.020

serverless pipeline



AWS CodePipeline

With AWS CodePipeline, there are no upfront fees or commitments. You pay only for what you use. AWS CodePipeline costs \$1 per active pipeline* per month. To encourage experimentation, pipelines are free for the first 30 days after creation.

* An active pipeline is a pipeline that has existed for more than 30 days and has at least one code change that runs through it during the month. There is no charge for pipelines that have no new code changes running through them during the month. An active pipeline is not prorated for partial months.

troubleshooting CodeBuild

- most likely permission related

troubleshooting CodeBuild

- most likely permission related
- uses ECS containers

troubleshooting CodeBuild

- most likely permission related
- uses ECS containers
- ECS passes AWS credentials to container differently to EC2 and Lambda

troubleshooting CodeBuild

- most likely permission related
- uses ECS containers
- ECS passes AWS credentials to container differently to EC2 and Lambda
- HTTP request to get AWS credentials can be slow on first request