Replicating Bitbucket Pipelines on your laptop for local debugging

April 17, 2020 BITBUCKET

This post was written by Bitbucket user Ayush Sharma.

Bitbucket Pipelines is one of my favourite CI/CD tools, and I use it pretty heavily daily. Given the full range of use-cases available for Pipelines, I have to frequently diagnose and debug new issues, and this process of debugging starts with being able to replicate the problems in my local environment quickly.

I'm going to walk you through the process of replicating a Serverless deployment locally. For this exercise, I'll use this **Serverless deployment as a reference**.

Clone your commit

Like all good CI/CD build tools, Pipelines happily gives you the actual commit-ID which triggered a particular build. In the Pipelines dashboard, the link to the commit-id is at the top left. In my case, the commit-id is `422c59b6f54c1a4e8xxxxxxxxxx`. So once I've cloned my repository locally, I'm going to checkout this commit.

```
git checkout 422c59b6f54c1a4e8xxxxxxxxxx
```

Checking out the same commit-id as the Pipelines build is vital. You don't want to check out the branch, only to find that another developer updated it right after your build. Remember that a git commit-id is a unique pointer to a change, so use it as intended and make sure you and your Pipelines are on the same commit.

Launch your Pipelines container

With the clone and checkout complete, go to your clone directory and launch the same Docker container as specified in your Pipelines.

To begin, `cat` your `bitbucket-pipelines.yml` to identify the container you're using.

```
image: node:11.13.0-alpine

pipelines:
  branches:
    master:
      - step:
          caches:
            - node
          script:
            - apk add python3
            - npm install -g serverless
            - serverless config credentials --stage dev --provider aws --key ${key} --secret
${secret}
            - serverless deploy --stage dev
```

Launch the `node:11.13.0-alpine` container and mount your current directory:

```
docker run -v `pwd`:/mycode -it node:11.13.0-alpine /bin/sh
```

Once the command runs successfully, you'll be in the Docker shell. Change to the mounted directory to ensure the code is present.

```
cd /mycode
```

Prepare your build locally

The hard part is already over! Now it's just a matter of executing the same build steps in this container. Only this time, we're on our laptop instead of on Atlassian's cloud!

In my container shell, I'm going to execute these commands:

```
/mycode
```

I'm skipping the serverless config steps because I won't be deploying anything from my laptop. At least not directly, and not without a PR

Running diagnostics

To debug my Serverless deployment, I find two things handy.

First, Serverless provides a debug-mode we can enable by setting an environment variable.

```
/mycode
```

With that done, we'll package our Serverless deployment without actually deploying it to the cloud. Sounds interesting, right? The command is as follows:

```
/mycode # serverless package --package /tmp/myserverlesspackage
```

This command will create the CloudFormation package for our project and dump the files inside /tmp/myserverlesspackage. Let's go there now and see what we have:

```
/mycode # cd /tmp/myserverlesspackage
/mycode

cloudformation-template-update-stack.json
cloudformation-template-create-stack.json
serverless-state.json
...
```

Nice! Our Serverless package built successfully. As you can see, our CloudFormation template is right there.

What we've done so far might now seem like much immediately, but we have quite a lot of diagnostic information already:

1. If there were any errors during the package step, then you've already eliminated Pipelines and everything that happens after it as a potential culprit. Narrowing down on problem areas is critical to diagnosing issues.
2. We can build the previous version of our project and compare the sizes of the CloudFormation templates, which can be diagnostically significant. This process is a lot harder to do on Bitbucket, especially if your build is continuously failing.
3. Replicating the environment locally also allows us to play with container versions. If you've been changing container versions and deploying Pipelines to test them, this will be a huge time-saver.
4. You can now also check the memory and CPU consumption of your build container in a controlled environment. If your Pipelines have been complaining of resource constraints, you now have a way to test it.

Local debugging avoids the friction of triggering your Pipeline every time you make a change.

Which means easier debugging, better debugging, and overall happier engineers.

Author bio: *Ayush Sharma is a software engineer specializing in infrastructure and automation. In his free time he enjoys hot tea and a good book.*

This post was originally posted in ***Ayush Sharma's Notes****.*

ABOUT THIS ARTICLE

AYUSH SHARMA

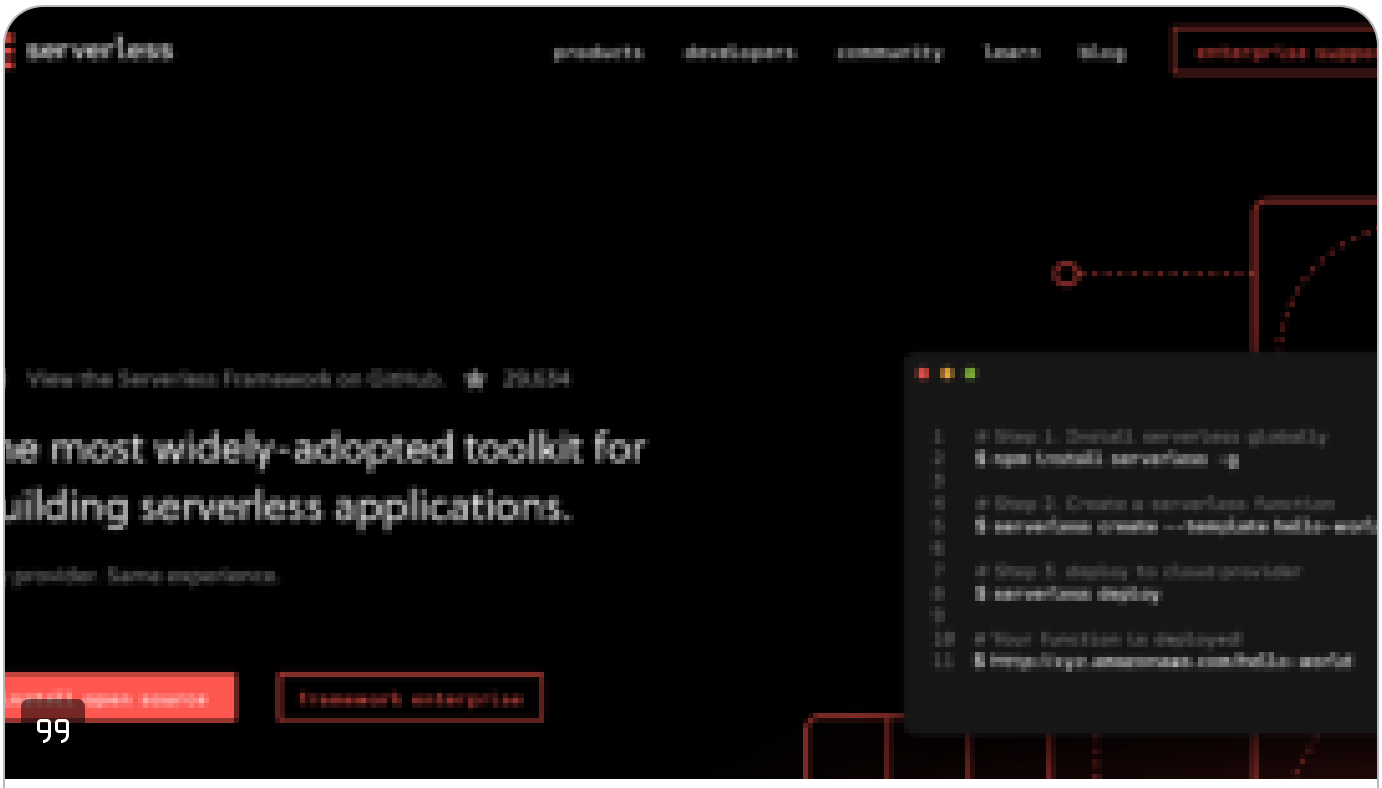
Subscribe for more

Inside **Atlassian**

Get insights like this in your inbox

Sign up now

Related content



ARTICLE IN BITBUCKET

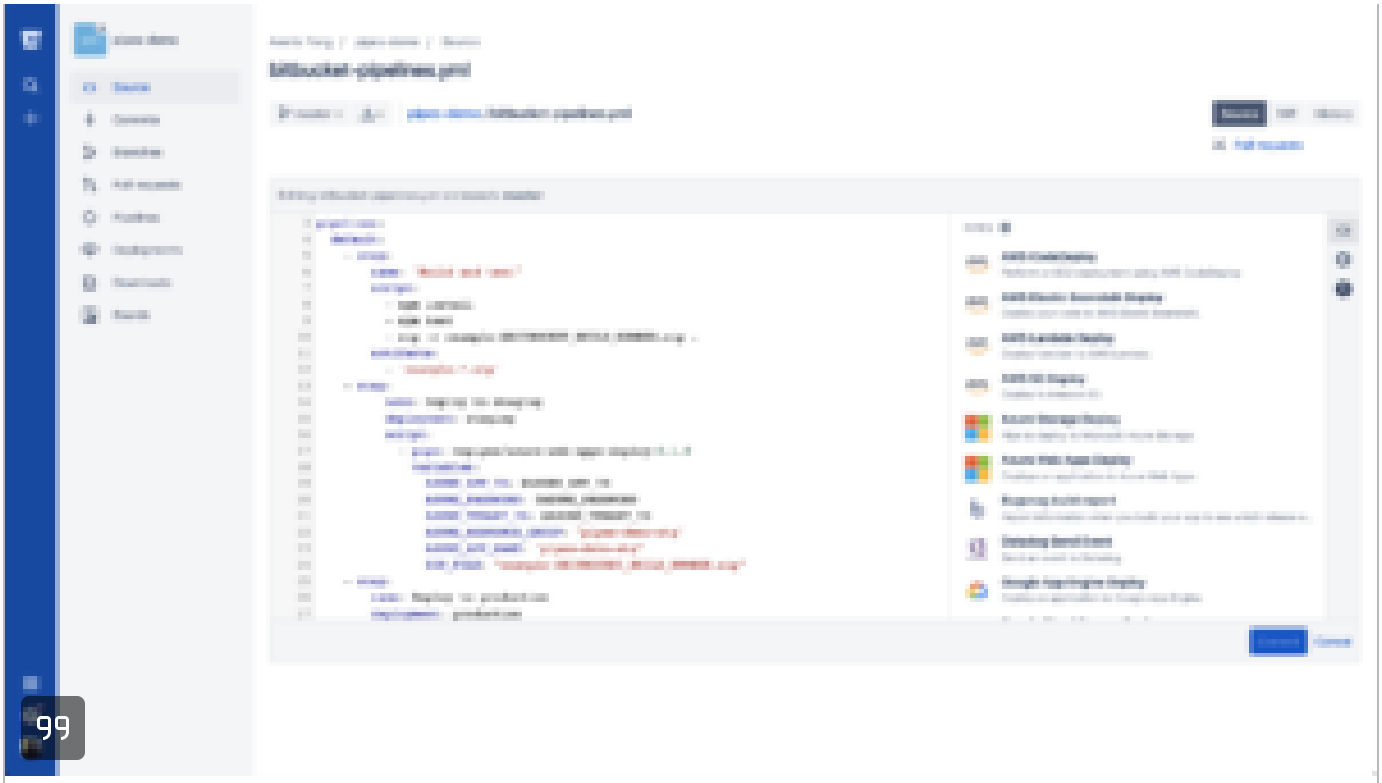
Automating Serverless framework deployments using Bitbucket Pipelines

Bitbucket now offers pre-built Pipes to deploy Lambdas to AWS. But what if we want to deploy an entire Serverless stack?

ARTICLE IN BITBUCKET

Rovo Chat in Bitbucket now understands your Pipelines

Why did your build fail? Ask Rovo, get a clear answer, and even a way to fix it, from anywhere in Bitbucket Pipeline debugging is one of the most common and most painful parts of the development workflow. In our Atlassian research: AI...



ARTICLE IN BITBUCKET

Meet Bitbucket Pipes. 30+ ways to automate your CI/CD pipeline

The democratizing nature of DevOps has seen the responsibility of building and managing CI/CD pipelines transition from specialized release engineers to developers. But automating a robust, dependable CI/CD pipeline is tedious work....

See all products →

Resources

Technical support

Purchasing & licensing

Atlassian Community

Team Playbook

Knowledge base

Marketplace

My account

Create support ticket →

Learn

Partners

Training & certification

Documentation

Developer resources

Enterprise services

See all resources →