CI with Cassandra, Portworx, Gitlab and DC/OS

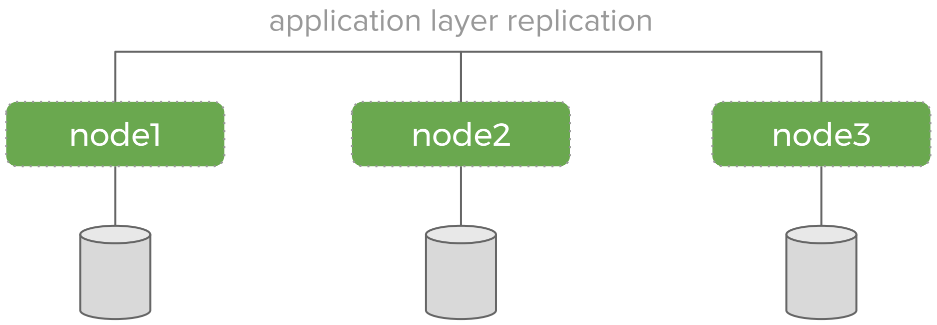
Running Cassandra on Portworx volumes.

Background

Background

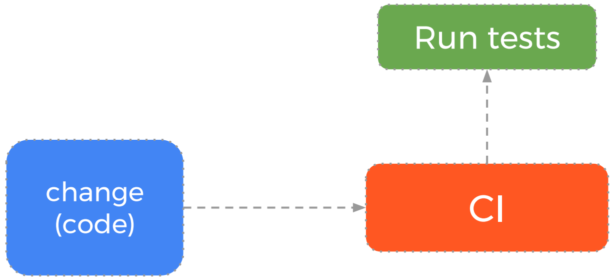
Running distributed databases such as Cassandra give you a fault tolerant solution to storage.

You can safely use local disk for your Cassandra cluster because if one of the nodes dies, even though it takes the disk with it - the cluster has replicated all of the data at the application layer.



Continuous Integration

CI means we can be sure that changes to our code do not break the system.



A developer changes some code and does $ git push. The CI system runs the test suite and you get the feedback that is most important:

"did the change the just happened break anything?"

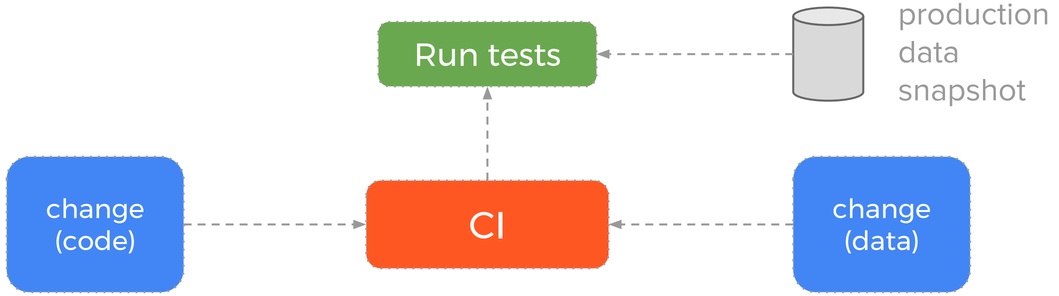
Data changes too

It turns out that developers making code changes is not the only change we need to account for - **data changes too**.

In an ideal world - we would be constantly testing against production data. This means we catch errors that come from production data we did not anticipate in our test suite.

Bidirectional CI

To solve this problem - we could continuously test against the latest production data:



If only we had a tool that could safely snapshot production data and present this volume to a Docker container so we can safely run our test-suite against it.

portworx

Portworx

Using Cassandra alone, this would be a challenge. Instead, we could use Portworx Developer Edition to create storage volumes for our Cassandra containers to use.

Then we can take snapshots of the production data and use the snapshot to run our test suite. Compared to using Cassandra alone, this means:

- the test suite has production data only seconds old
- we catch errors that result in data we had not anticipated in our tests
- our test suite can safely write to the database because it's a snapshot

portworx

Demo Time

Demo Time

In the demonstration video - the following scenario unfolds.

- We deploy a change to our guest book application
  - The test suite passes and we go home to bed
- The next morning everything is broken
  - A user had entered a strange utf-8 character
  - Even though our tests passed - the app is still broken in production
- We use Portworx snapshots and Gitlab CI
  - We configure our tests to use a portworx snapshot
  - Now our tests catch errors that result in data we had not anticipated in our tests

Guestbook Application

Our guestbook application is running smoothly. We are running our app on DC/OS and have a Cassandra database so we don't need to worry about if a node dies.

We notice that the API is returning an unfavourable data format and decide to make a change. We make the change and push the code to our CI server. The tests pass and we check the results in the deployed application.

Everything is working and we have a good feeling about doing devops right. We head off to bed and plan out the next product features.

portworx

Everything is broken

The next morning we notice the site is broken. Nothing is loading and we start to think:

“but all my tests passed last night - what could have changed?”

Everything is broken

We start to dig around and notice that a user had typed a bad character and our code breaks. Our test suite inserts dummy data into a Cassandra database and that is how our test suite did not pick up this problem - it only existed in the production system.

The realisation that we need to test against both the latest code **and** production data hits us.

Portworx snapshots

Thankfully - we had installed our Cassandra database onto a Portworx volume. This makes it easy to take a snapshot of our production data and use it for our tests.

We modify our tests and CI config to use this volume when running the test suite. This time - when we push the code, the test-suite catches the error and now everything makes sense.

We fix the error, the test suite passes and our app is back up and online.

portworx

CI with Cassandra, Portworx, Gitlab and DC/OS

Visit the Portworx website to find out more!