



AARHUS UNIVERSITET

# Microservices and DevOps

DevOps and Container Technology

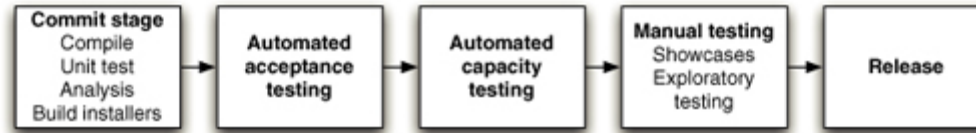
Build Pipelines

Henrik Bærbak Christensen

- *Deployment Pipeline*: An automated implementation of your application's build, deploy, test, and release process.

Humble et al.

Figure 1.1 The deployment pipeline

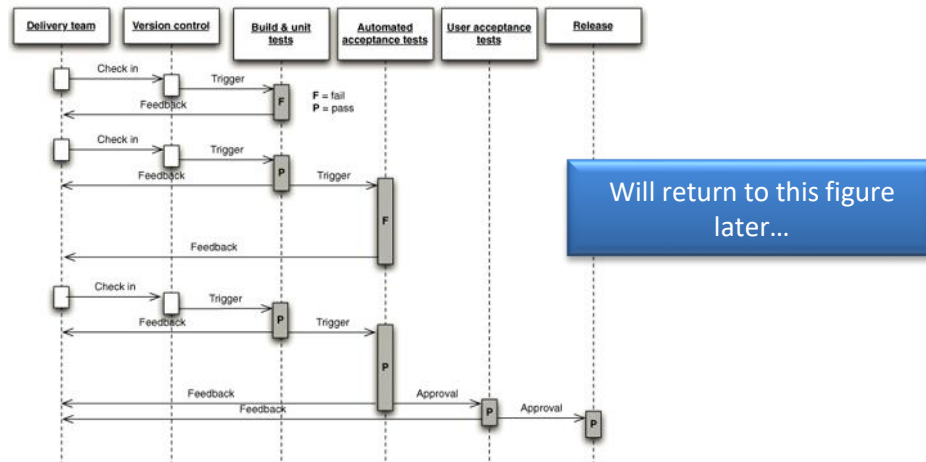


- Every change that is made in any artefacts of the application *triggers* the creation of a new instance of the pipeline.
  - Series of tests, each more demanding, each giving confidence in that *it will work*. Passing all tests means ready for release.

# Pipeline

- Newman (as usual) waves his hand a bit...
  - To have different stages in your build, creating what is known as a build pipeline [p 107]*
  - Fast test followed by slow tests
    - Why run the slow ones if the fast tests fail?

- Pattern: *Fail fast*



# Pipeline

- Pipeline = stages of build, test, deploy, and release
- Modern build tools have aspects of this
  - Gradle 'jar' = compile, test, build deployment unit
  - The pipeline is predefined by gradle
- And – you could easily handcode (parts) of it!
  - A bash script that does gradle jar, docker build, docker push
- Of course, better with good tool support

# Tech Choice?

- Lots of Build servers out there
  - Jenkins, Hudson, GoCD, Bamboo, Concourse, ...
- Git and Bitbucket have their own system
  - *Rather* well integrated with the SCM system!!!
    - *Had a lot of issues with Concourse and Jenkins about checking out of a private repo...*
- Bitbucket in MSDO because...
  - History – I started there...
  - **TestContainers work well within BitBucket !**
- You are free to pick any you like...
  - But I can help more if you choose BitBucket 😊

## Key concepts

A pipeline is made up of a set of steps.

- Each step in your pipeline runs a separate Docker container. If you want, you can use different types of container for each step, by selecting different images.
- The step runs the commands you provide in the environment defined by the image.
- A single pipeline can have up to 10 steps.

Infrastructure-as-code  
*bitbucket-pipelines.yml*  
(YAML format of course)

<https://support.atlassian.com/bitbucket-cloud/docs/get-started-with-bitbucket-pipelines/>

# Key Concepts

**pipelines:**

**default:**

**step:**

**script:**

**branches:**

**staging:**

**step:**

**feature/\*:**

**step:**

# Example Pipeline

- From Bitbucket sample code

```
# You can specify a custom docker image from Docker Hub as your build environment.
image: maven:3.3.9

pipelines:
  default:
    - step:
        caches:
          - maven
        script: # Modify the commands below to build your repository.
          - mvn -B verify # -B batch mode makes Maven less verbose

        services:
          - mongo

definitions:
  services:
    mongo:
      image: mongo
```

```
image: openjdk:8
pipelines:
  default:
    - step:
        script:
          - bash ./gradlew build
```

<https://bitbucket.org/bitbucketpipelines/pipelines-guide-java/src/master/bitbucket-pipelines.yml>

# Strong Hint for SkyCave

- Potential first step in solution

```
image: openjdk:11

pipelines:
  default:

    - step: # === Unit test
      name: Unit Test
      caches:
        - gradle

      script:
        - bash ./gradlew test jacocoRootReport
```

- Huh? Why './gradlew' ???

- From bottom up is expensive wrt dependencies
  - Download everything every time ☹️

## What is dependency caching?

Most builds start by running commands that download dependencies from the internet, which can take a lot of time for each build. As the majority of dependencies stay the same, rather than download them every time, we recommend downloading them once into a cache which you can reuse for later builds.

- Predefined caches:

|        |                                   |                               |
|--------|-----------------------------------|-------------------------------|
| gradle | <pre>1 caches: 2   - gradle</pre> | <code>~/.gradle/caches</code> |
|--------|-----------------------------------|-------------------------------|

# Commit Triggered

Henrik Bærbak Christensen / cave

## Pipelines

[What's new](#)

[Schedules](#)

[Caches](#)

[Usage](#)



All branches

Status



Trigger type



Only mine

OTHER BRANCHES

master **MAIN BRANCH**

f20-solution

dev

msdo-dev

msdo-dev2

digiinno19

msdo-broker-intro

itminds-course

| Pipeline                                                                                                                                                                                                                                                                                                                                                                           | Status                                                                                          | Started           | Duration     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|-------------------|--------------|
| #10  Updated docs<br>Henrik Bærbak Christensen  b443953  f20-solution                                                           |  In progress | a few seconds ago | .            |
| #9  Added deployment stage to pipeline<br>Henrik Bærbak Christensen  8eeb3b3  f20-solution                                      |  Successful  | an hour ago       | 3 min 1 sec  |
| #8  Added docker login to allow test containers to access private repo i...<br>Henrik Bærbak Christensen  ec2b531  f20-solution |  Successful  | 2 hours ago       | 1 min 29 sec |
| #7  Added integration/service tests to the pipeline<br>Henrik Bærbak Christensen  78512fa  f20-solution                         |  Failed      | 2 hours ago       | 1 min 21 sec |
| #6  Merged dev into f20<br>Henrik Bærbak Christensen  9aea988  f20-solution                                                     |  Successful  | 2 hours ago       | 29 sec       |

# Pipeline Log

Henrik Bærbak Christensen / cave / Pipelines

✓ #10 Rerun

🔗 b443953 Updated docs

🔗 f20-solution

🕒 2 min 33 sec 📅 5 minutes ago 

### Pipeline

- ✓ Unit Test 112 tests passed • 42s
- ✓ Service tests 16 tests passed • 1m 08s
- ✓ Image Deployment 42s

```
Build docker +

Build setup 6s >
  VERSION=$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 32 | head -n 1 | xargs echo)
  LATEST=$(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 32 | head -n 1 | xargs echo)
  echo $(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 32 | head -n 1 | xargs echo) > docker tags --repository "f20-solution/$VERSION" --password-stdin
  docker build --tag "f20-solution/$VERSION" .
  echo The image name is $(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 32 | head -n 1 | xargs echo)
  docker build -t f20-solution/$VERSION --tag f20-solution/$VERSION .
  docker tag f20-solution/$VERSION f20-solution/$LATEST
  docker push f20-solution/$VERSION
  docker push f20-solution/$LATEST
  git tag -a "image_$VERSION" -m "Deployed docker image f20-solution/$VERSION"
  git push origin "image_$VERSION"
Build teardown <1s >
```

# Image Release

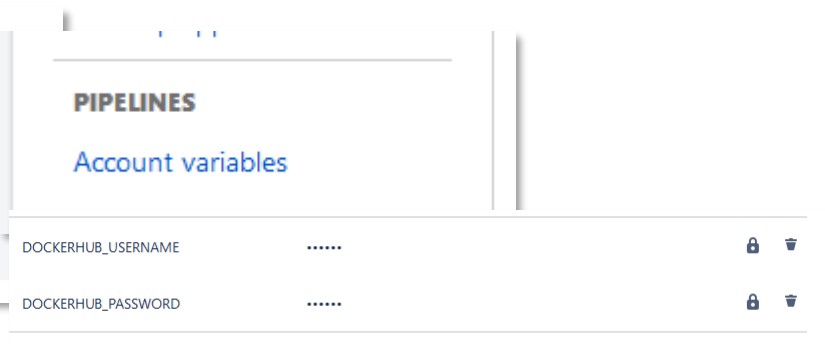
- ... means *pushing the built image to Docker hub*
- ... means having the credentials to Docker hub
- So you need to store the credentials in BitBucket
  - **Do not have them in the script!!!**

## Authenticate when pushing to a registry

To push images to a registry, you need to use `docker login` to authenticate prior to calling `docker push`. You should set your username and password using [variables](#).

For example, add this to your pipeline script:

```
docker login --username $DOCKER_USERNAME --password $DOCKER_PASSWORD
```



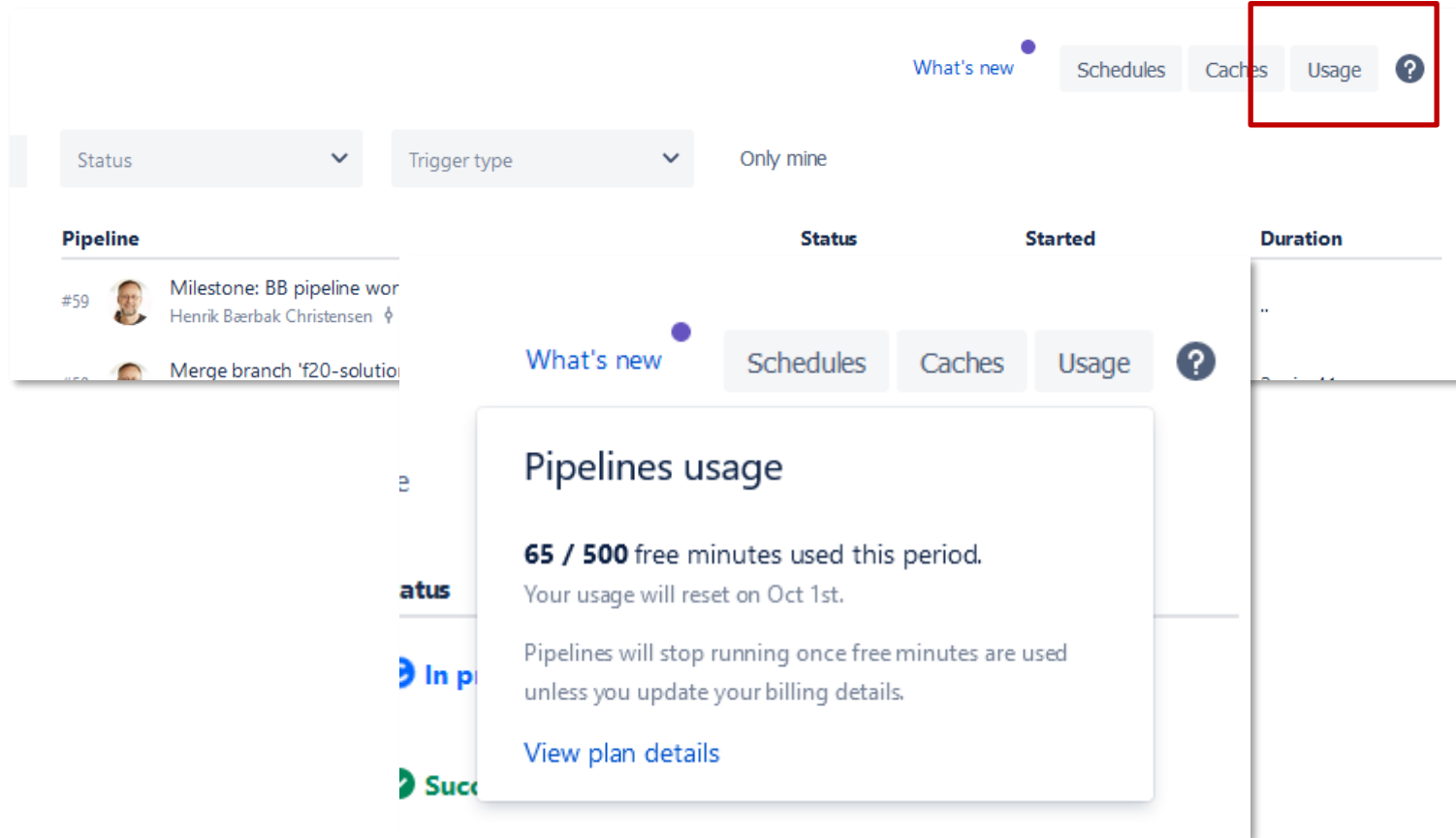
- And I always spend 15 minutes trying to find the variables 😞

- Free offer limit is 50 build minutes per month

| Pricing details           |                           | <a href="#">Hide details -</a> |
|---------------------------|---------------------------|--------------------------------|
| <b>Free</b>               | 50 min / month included   |                                |
| <b>Standard</b>           | 500 min / month included  |                                |
| <b>Premium</b>            | 1000 min / month included |                                |
| <b>Additional minutes</b> | \$10 / month for 1000 min |                                |

- So...
  - Keep the pipeline in your 'deploy' branch only!
    - In 'dev'/'master' you will run out of build time very soon...
  - Or – get an 'academic license' – I get 500 minutes for free...

# Review Usage



What's new Schedules Caches **Usage** ?

Status Trigger type Only mine

| Pipeline                                                                                                                                      | Status | Started | Duration |
|-----------------------------------------------------------------------------------------------------------------------------------------------|--------|---------|----------|
| #59  Milestone: BB pipeline wor<br>Henrik Bærbak Christensen |        |         |          |
| #58  Merge branch 'f20-solution                              |        |         |          |

**Pipelines usage**

**65 / 500** free minutes used this period.  
Your usage will reset on Oct 1st.

Pipelines will stop running once free minutes are used  
unless you update your billing details.

[View plan details](#)

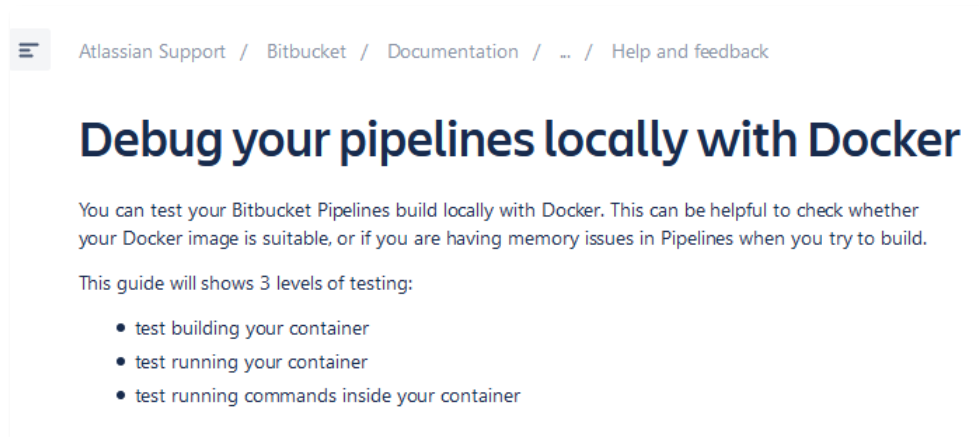


# Take Small Steps

- Creating your first pipeline has a somewhat steep learning curve...
- Tutorial material is, well, ok.
  - Bit convoluted...
- Take small steps
  - Lots of commits, lots of waiting for pipeline to start, but debugging is not funny

# Small Steps

- Maybe have a look at (though I have not tried it)



Atlassian Support / Bitbucket / Documentation / ... / Help and feedback

## Debug your pipelines locally with Docker

You can test your Bitbucket Pipelines build locally with Docker. This can be helpful to check whether your Docker image is suitable, or if you are having memory issues in Pipelines when you try to build.

This guide will show 3 levels of testing:

- test building your container
- test running your container
- test running commands inside your container

# Integration Tests in BitBucket

- TestContainers **works well** in Bitbucket pipelines !

## Testcontainers

Home

Quickstart ▾

Features ▾

Modules ▾

Test framework integration ▾

System Requirements ▲

General Docker requirements

Continuous Integration ▲

Patterns for running tests  
inside a Docker container

CircleCI 2.0

Drone CI

GitLab CI

Bitbucket Pipelines

Windows Support

Recommended logback  
configuration

Image Registry rate limiting

Getting help

Contributing ▾

## Bitbucket Pipelines



To enable access to Docker in Bitbucket Pipelines, you need to add `docker` as a service on the step.

Furthermore, Ryuk needs to be turned off since Bitbucket Pipelines does not allow starting privileged containers (see [Disabling Ryuk](#)). This can either be done by setting a repository variable in Bitbucket's project settings or by explicitly exporting the variable on a step.

In some cases the memory available to Docker needs to be increased.

Here is a sample Bitbucket Pipeline configuration that does a checkout of a project and runs maven:

```
image: maven:3.6.1

pipelines:
  default:
    - step:
        script:
          - export TESTCONTAINERS_RYUK_DISABLED=true
          - mvn clean install
        services:
          - docker
  definitions:
    services:
      docker:
        memory: 2048
```

# Summary

- *Deployment Pipeline*: An automated implementation of your application's build, deploy, test, and release process.
- *Build Pipeline*: An automated implementation of your application's build, deploy, test, ~~and release~~ process.
- Uses the *Fail fast* pattern
- Tool supported – triggered by (certain) SCM commits