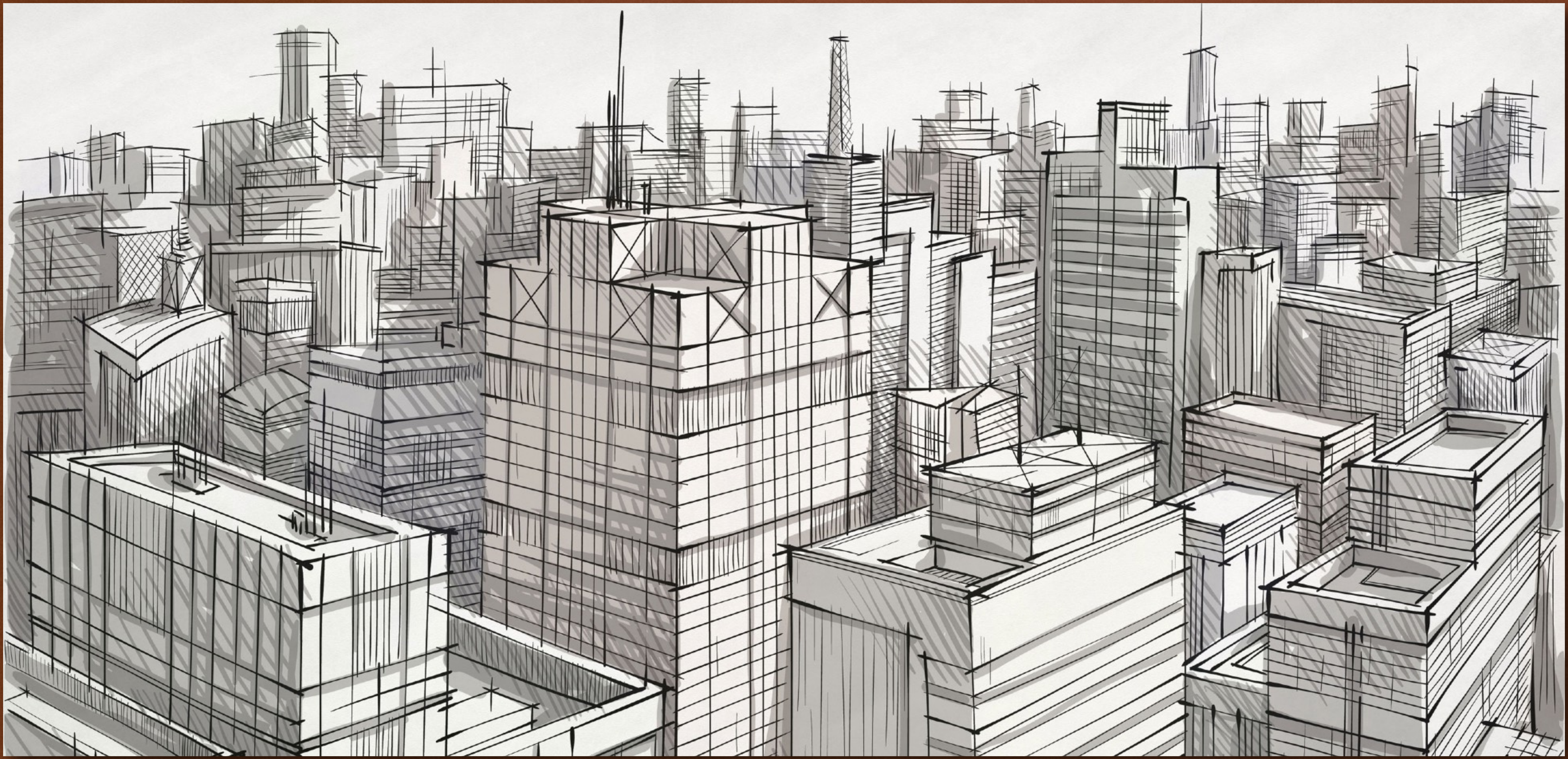
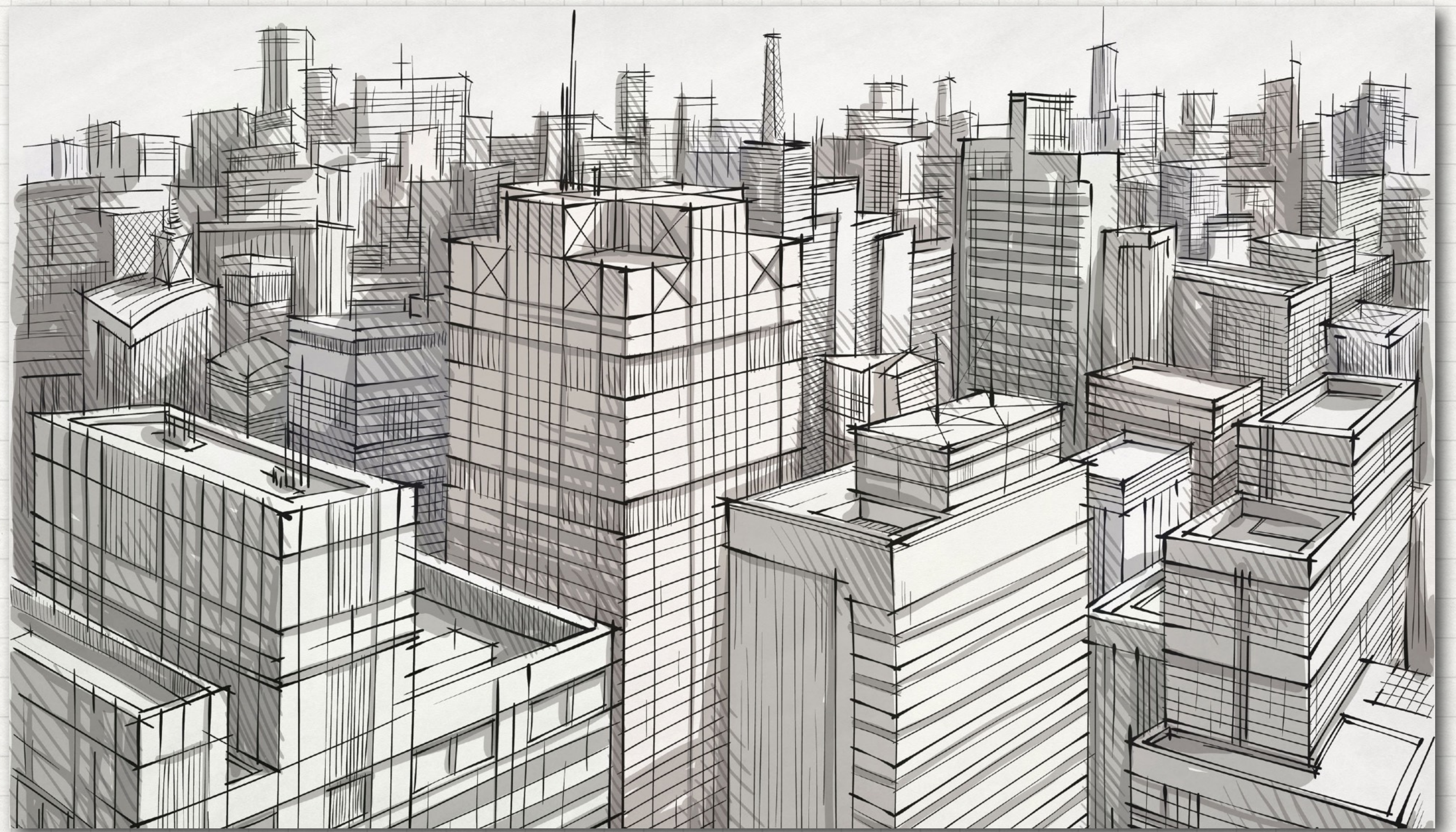
GIT WITH CI

DRONE.IO AND GITEA.IO FOR LIGHTWEIGHT CONTINUOUS INTEGRATION (CI)



BUILD STUFF WITH GIT AND CI

CONTINUOUS INTEGRATION (CI) IS THE PROCESS OF AUTOMATING THE BUILD AND TESTING OF CODE EVERY TIME A TEAM MEMBER COMMITS CHANGES TO VERSION CONTROL *1)*



1) https://www.visualstudio.com/learn/what-is-continuous-integration/

CI TOOLS

CI ALTERNATIVES

TRAVIS CI

- free only for Open Source projects
- no self-hosting (?)

CIRCLE CI

- limited self-hosting, free version only for 2 Ubuntu versions
- limited free plan
- only Github and Bitbucket integrations
- limited to specific languages (Go (Golang), Haskell, Java, PHP, Python, Ruby/Rails, Scala)

JENKINS CI

- rather complex to setup / define pipelines / workflows (but "Jenkins Blue Ocean" makes it a lot easier)
- may need a bunch of plugins to get desired functionality
- based on Java, higher system requirements

GITLAB CI

- installs / runs a bunch of bundled software packages (Postgres, Nginx, Prometheus, Ruby, Sidekiq, Docker Registry, Kubernetes support, ...)
- rather high system requirements

DRONE.IO

CONFIGURATION AS CODE. DOCKER NATIVE.

Drone is a lightweight, powerful continuous delivery platform built for containers.

Drone is packaged and distributed as a Docker image and can be downloaded from Dockerhub.

DRONE.IO

INSTALLATION

- http://docs.drone.io/installation/
- http://docs.drone.io/install-for-gitea/
- default storage engine is an embedded SQLite database, Mysql, Postgres supported
- "install" via Docker Compose - starts the "drone server" and a "drone agent", which is running the builds
- standalone or with proxy (Nginx, Apache, Caddy, ...) possible
- SSL, Letsencrypt supported

DRONE.IO

EXAMPLE WITH DOCKER-COMPOSE USING GITEA

docker-compose.yml

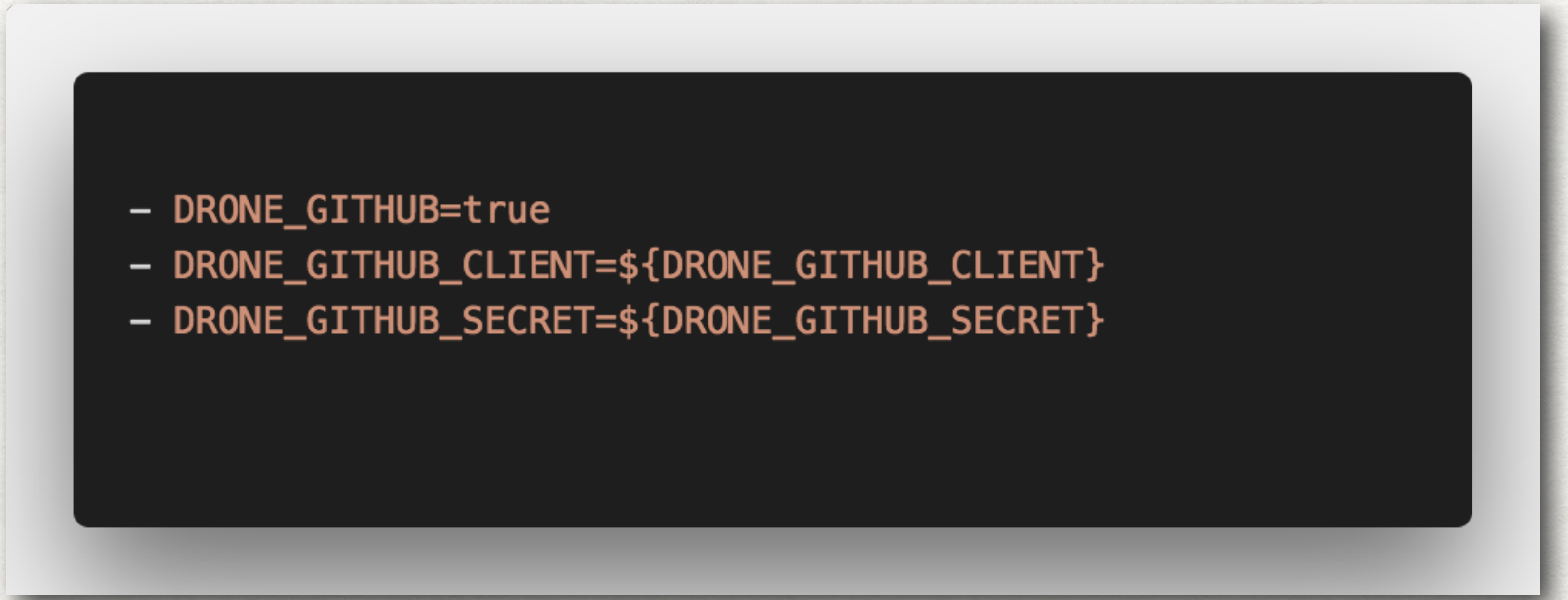
```yaml
version: '2'

services:
  drone-server:
    image: drone/drone:0.8

    ports:
      - 8000:8000
      - 9000
    volumes:
      - /var/lib/drone:/var/lib/drone/
    restart: "always"
    environment:
      - DRONE_OPEN=${DRONE_OPEN}
      - DRONE_HOST=${DRONE_HOST}
      - DRONE_GITEA=true
      - DRONE_GITEA_URL=${DRONE_GITEA_URL}
      - DRONE_SECRET=${DRONE_SECRET}
      - DRONE_ADMIN=smoises

  drone-agent:
    image: drone/agent:0.8

    restart: "always"
    depends_on:
      - drone-server
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    environment:
      - DRONE_SERVER=drone-server:9000
      - DRONE_SECRET=${DRONE_SECRET}
```

.env

```
DRONE_HOST=www.myserver.de
DRONE_GITEA_URL=http://www.my-git-server.de:1337/
DRONE_SECRET=abcde222222111
DRONE_OPEN=false
```

DRONE INTEGRATIONS

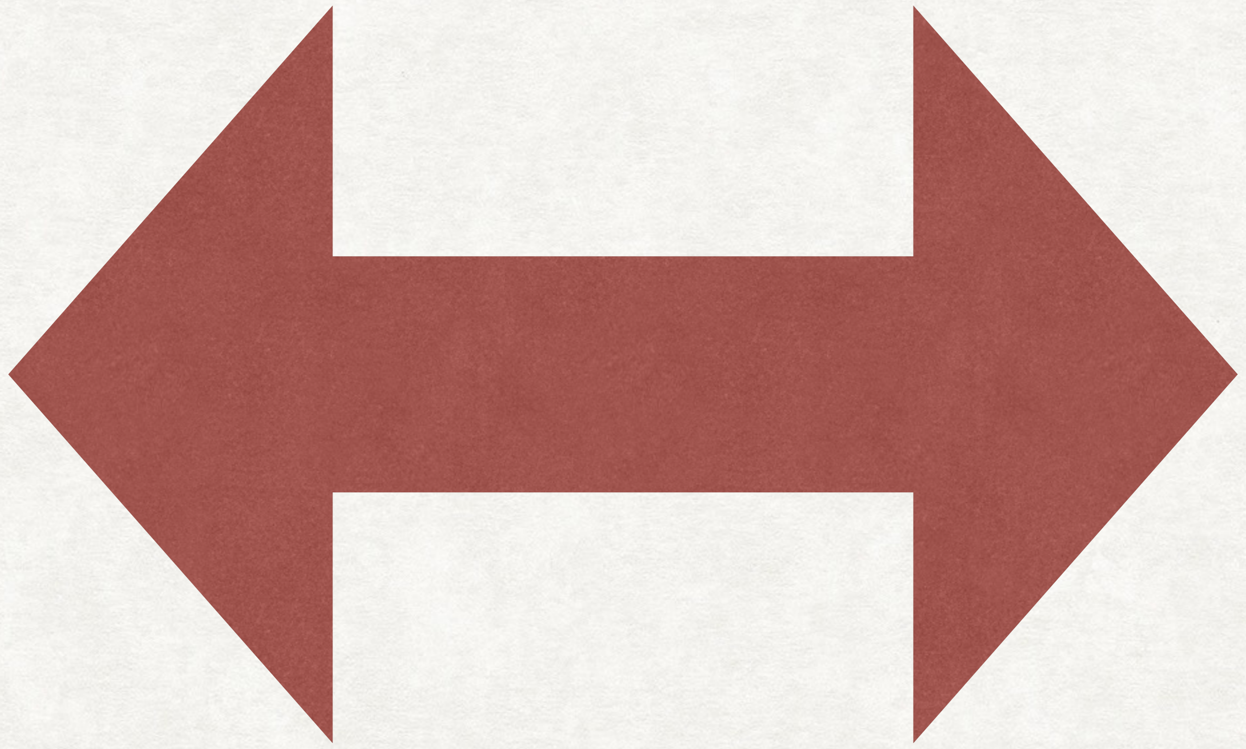
- Gitea / Gogs authentication via username / password (Gitea has no OAuth2 support)
- besides Gitea and Gogs, **Github, Gitlab, Bitbucket** etc. are also possible (mostly using OAuth2)



```
- DRONE_GITHUB=true
- DRONE_GITHUB_CLIENT=${DRONE_GITHUB_CLIENT}
- DRONE_GITHUB_SECRET=${DRONE_GITHUB_SECRET}
```

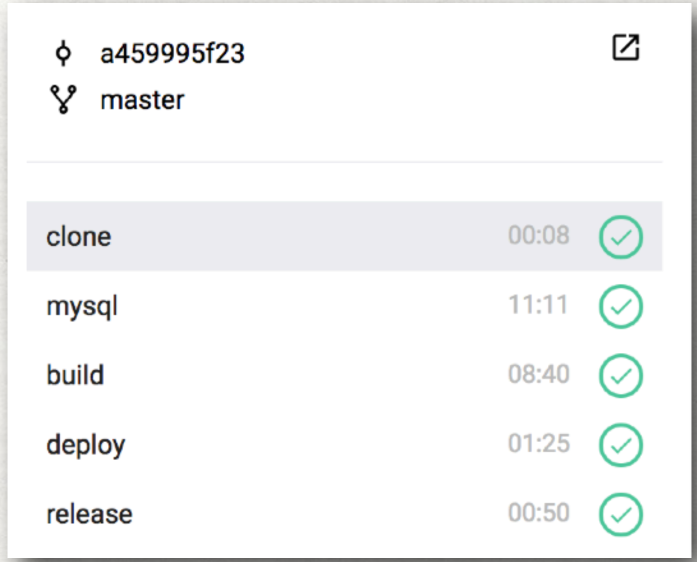
SCALING VIA AGENTS

- you can add more agents to increase the number of parallel builds
- you can also adjust the agent's DRONE_MAX_PROCS=1 environment variable to increase the number of parallel builds for that agent



PIPELINES

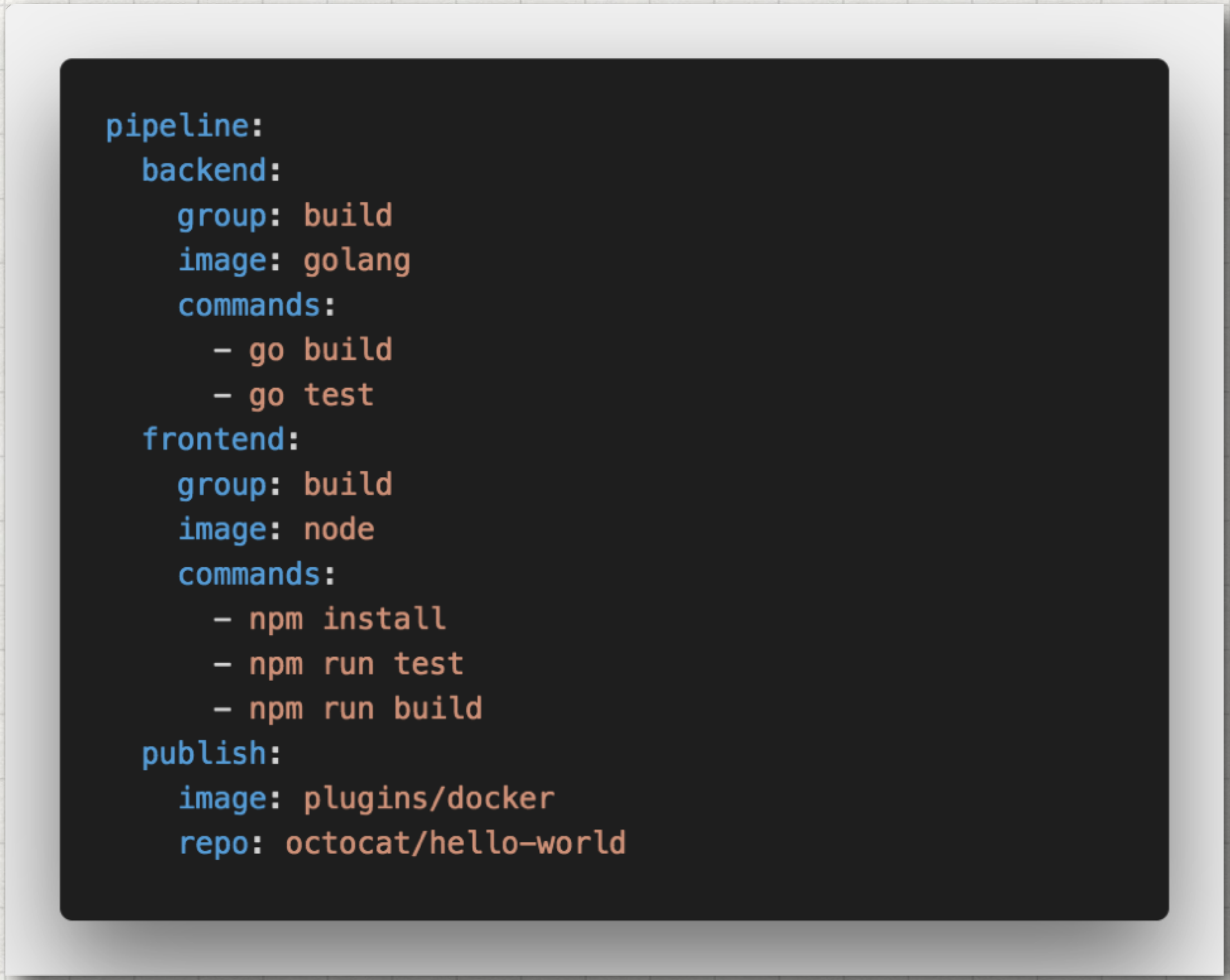
- define a list of steps to build, test and deploy your code
- pipeline steps are executed serially, in the order in which they are defined
- if a step returns a non-zero exit code, the pipeline immediately aborts and returns a failure status
- the names of the steps are completely arbitrary
- Drone supports parallel step execution
- parallel steps are configured using the group attribute



a459995f23
master

clone	00:08	✓
mysql	11:11	✓
build	08:40	✓
deploy	01:25	✓
release	00:50	✓

A PIPELINE WITH GROUPS

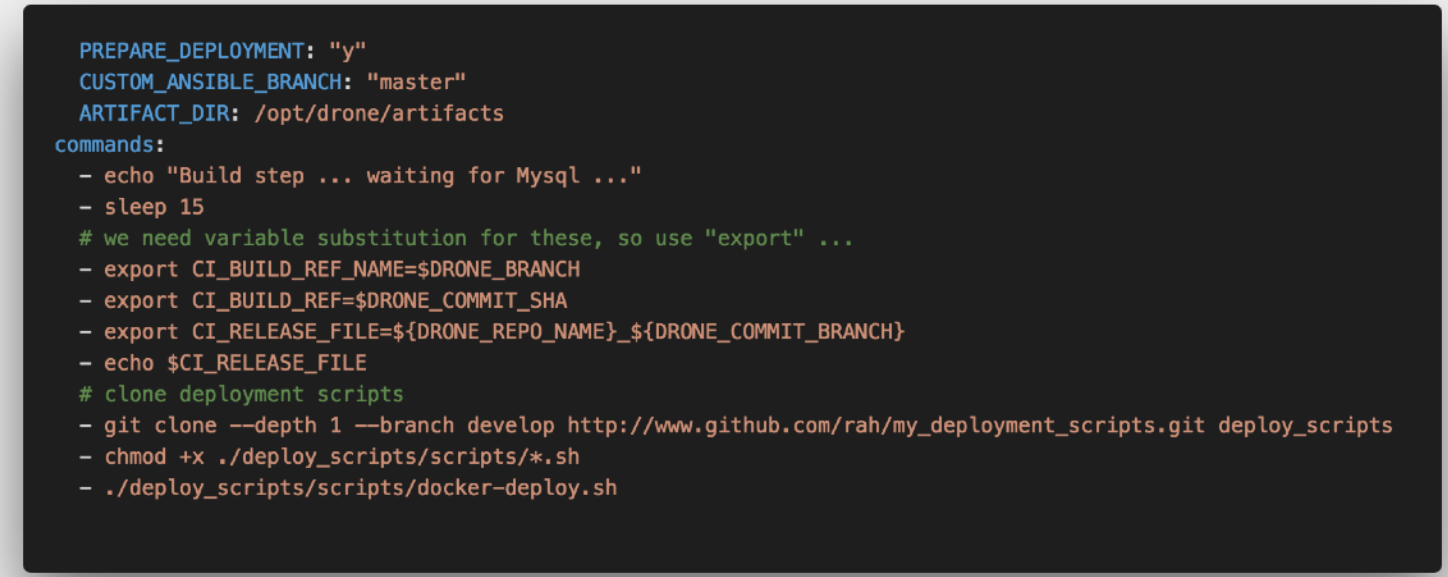


```
pipeline:
  backend:
    group: build
    image: golang
    commands:
      - go build
      - go test
  frontend:
    group: build
    image: node
    commands:
      - npm install
      - npm run test
      - npm run build
  publish:
    image: plugins/docker
    repo: octocat/hello-world
```

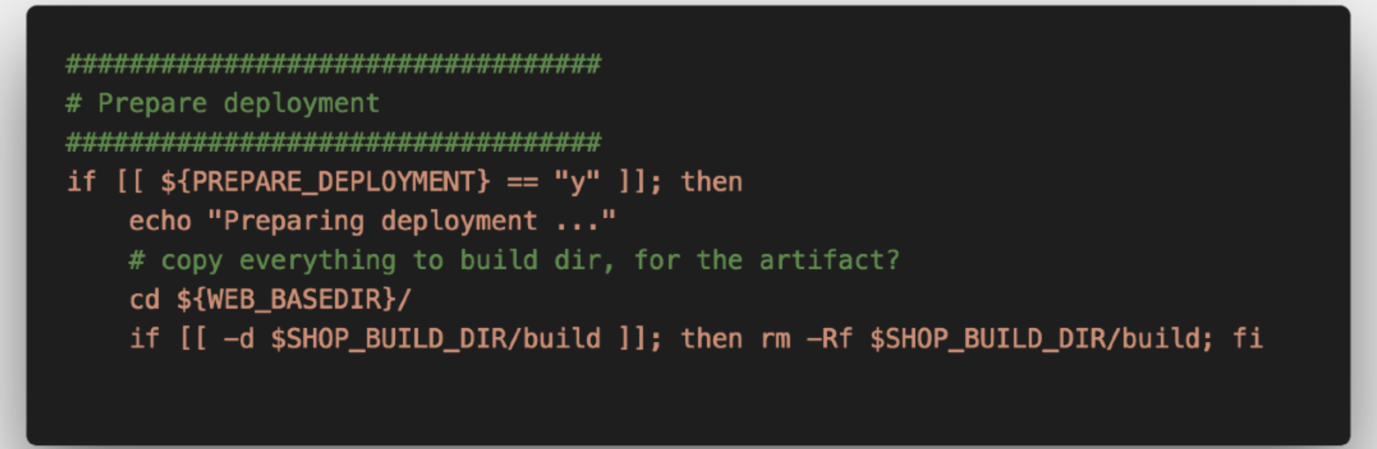
PIPELINE VARIABLES

- similar to e.g. Gitlab CI, you can **define variables** in your .drone.yml file which will be available in your docker containers or in succeeding bash scripts etc.

- tip: you can also clone additional repos to have a central place for e.g. deployment scripts - don't copy / paste dozens of bash lines to your pipelines' „commands" sections



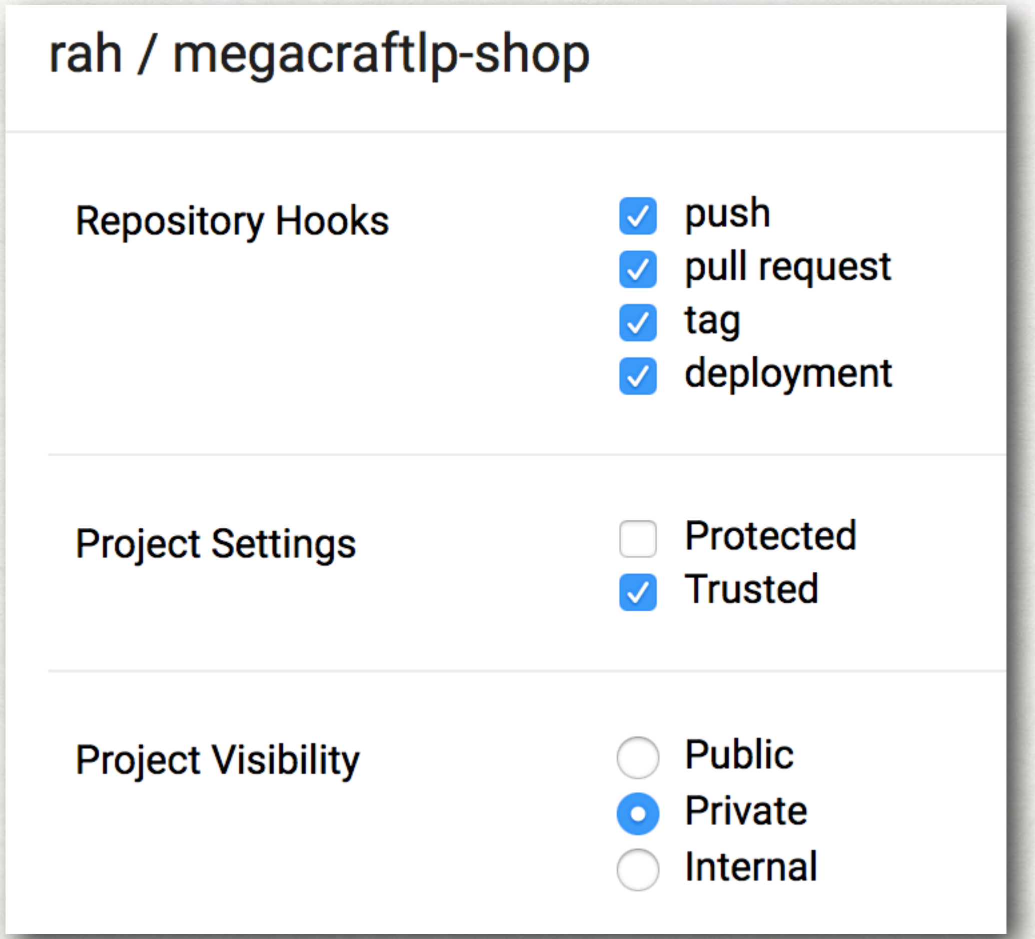
```
  PREPARE_DEPLOYMENT: "y"
  CUSTOM_ANSIBLE_BRANCH: "master"
  ARTIFACT_DIR: /opt/drone/artifacts
commands:
  - echo "Build step ... waiting for Mysql ..."
  - sleep 15
  # we need variable substitution for these, so use "export" ...
  - export CI_BUILD_REF_NAME=$DRONE_BRANCH
  - export CI_BUILD_REF=$DRONE_COMMIT_SHA
  - export CI_RELEASE_FILE=${DRONE_REPO_NAME}_${DRONE_COMMIT_BRANCH}
  - echo $CI_RELEASE_FILE
  # clone deployment scripts
  - git clone --depth 1 --branch develop http://www.github.com/rah/my_deployment_scripts.git deploy_scripts
  - chmod +x ./deploy_scripts/scripts/*.sh
  - ./deploy_scripts/scripts/docker-deploy.sh
```



```
##################################
# Prepare deployment
##################################
if [[ ${PREPARE_DEPLOYMENT} == "y" ]]; then
    echo "Preparing deployment ..."
    # copy everything to build dir, for the artifact?
    cd ${WEB_BASEDIR}/
    if [[ -d $SHOP_BUILD_DIR/build ]]; then rm -Rf $SHOP_BUILD_DIR/build; fi
```

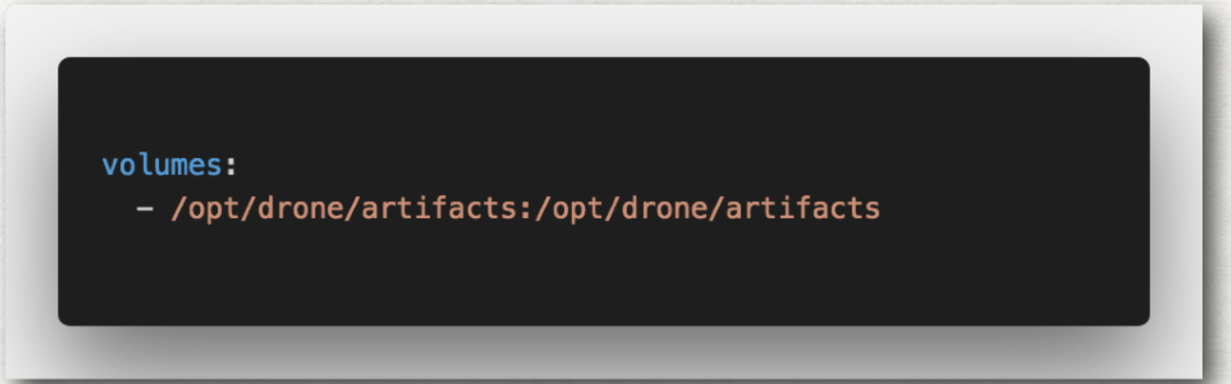
DOCKER VOLUMES

- only available to trusted repositories
- for security reasons should only be used in private environments



rah / megacraftlp-shop

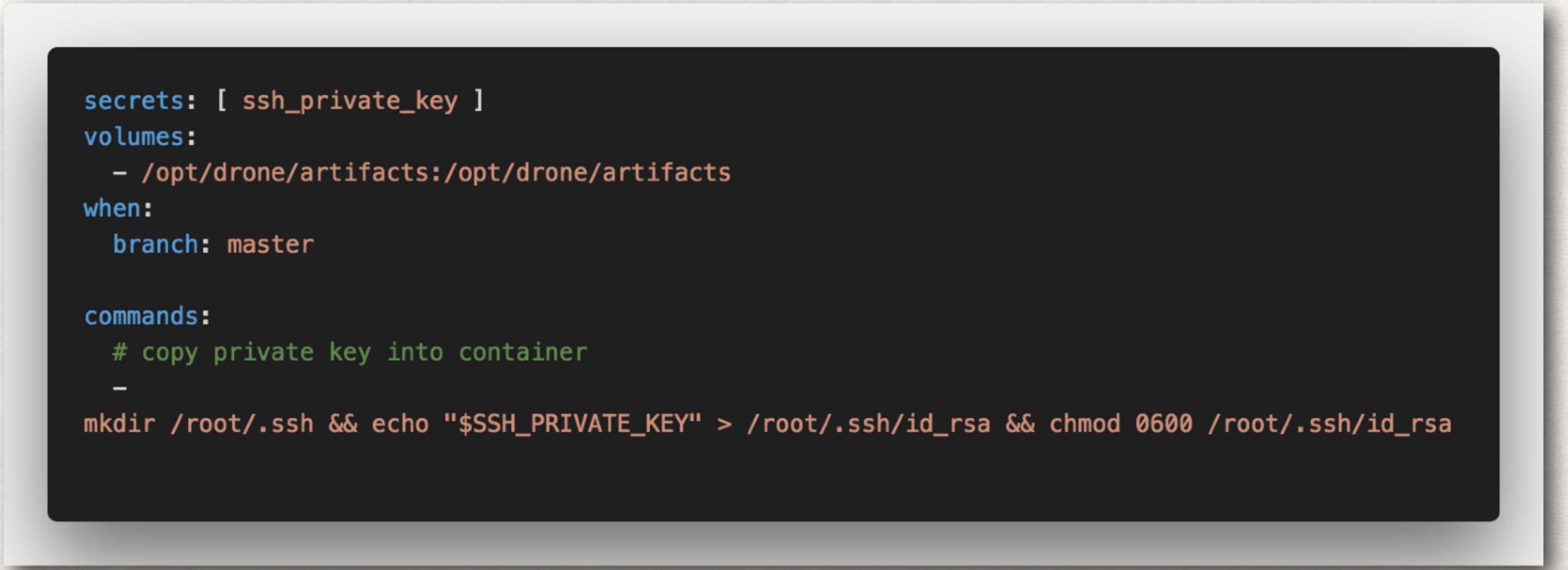
Repository Hooks	✓ push ✓ pull request ✓ tag ✓ deployment
Project Settings	☐ Protected ✓ Trusted
Project Visibility	○ Public ● Private ○ Internal



```
volumes:
  - /opt/drone/artifacts:/opt/drone/artifacts
```

SECRETS

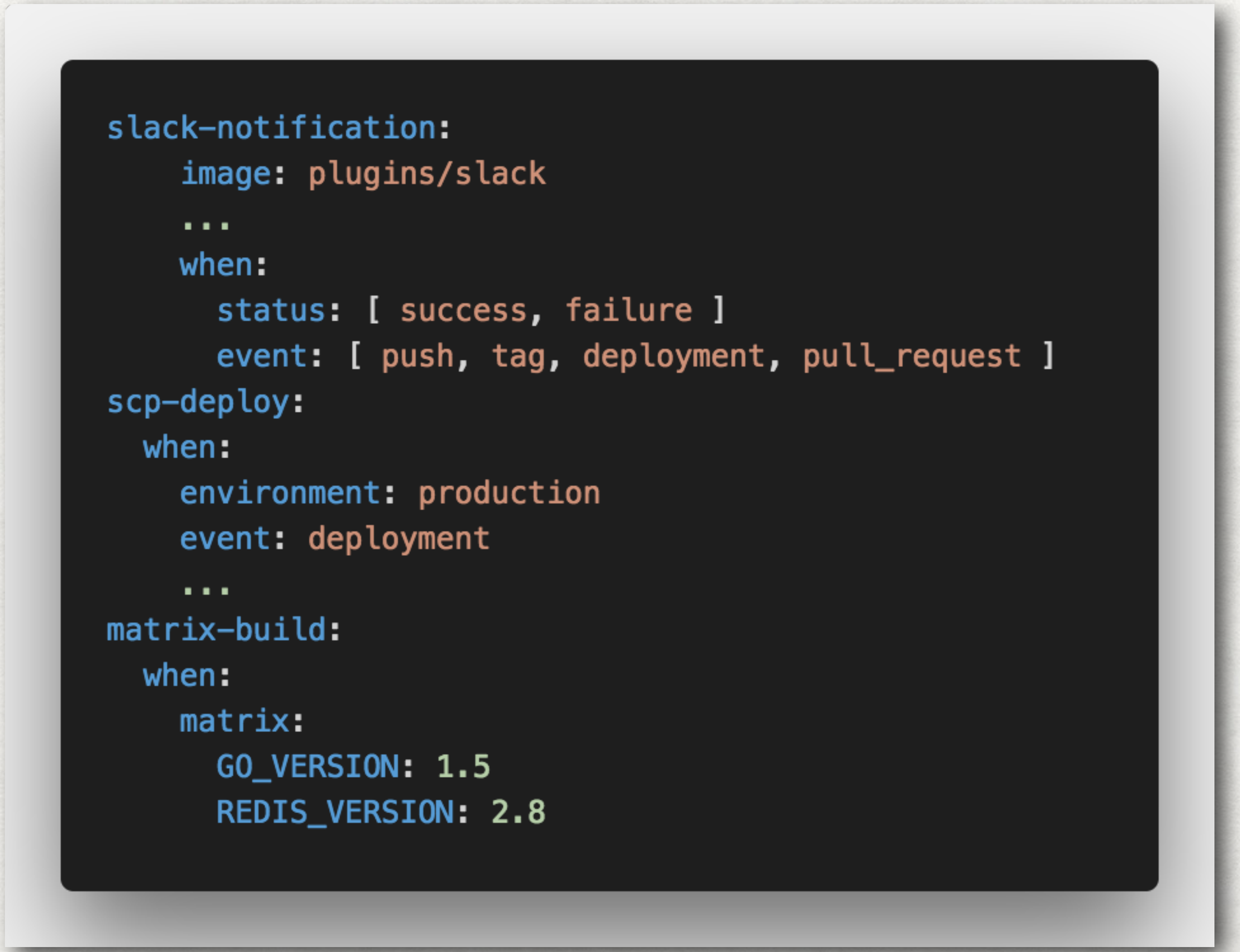
- Drone provides the ability to store named parameters external to the Yaml configuration file, in a central secret store
- the secrets are exposed to the plugin as uppercase environment variables
- drone secret add \
  -repository rah/megacraftlp-shop \
  -name ssh_private_key \
  -value @/opt/drone/id_rsa



```
secrets: [ ssh_private_key ]
volumes:
  - /opt/drone/artifacts:/opt/drone/artifacts
when:
  branch: master

commands:
  # copy private key into container
  -
mkdir /root/.ssh && echo "$SSH_PRIVATE_KEY" > /root/.ssh/id_rsa && chmod 0600 /root/.ssh/id_rsa
```

CONDITIONAL BUILDS

- Drone supports defining conditional pipelines and steps
- matrix builds are supported
- other conditions include status of builds, GIT events, environments or platforms as well as only for certain instances, e.g.

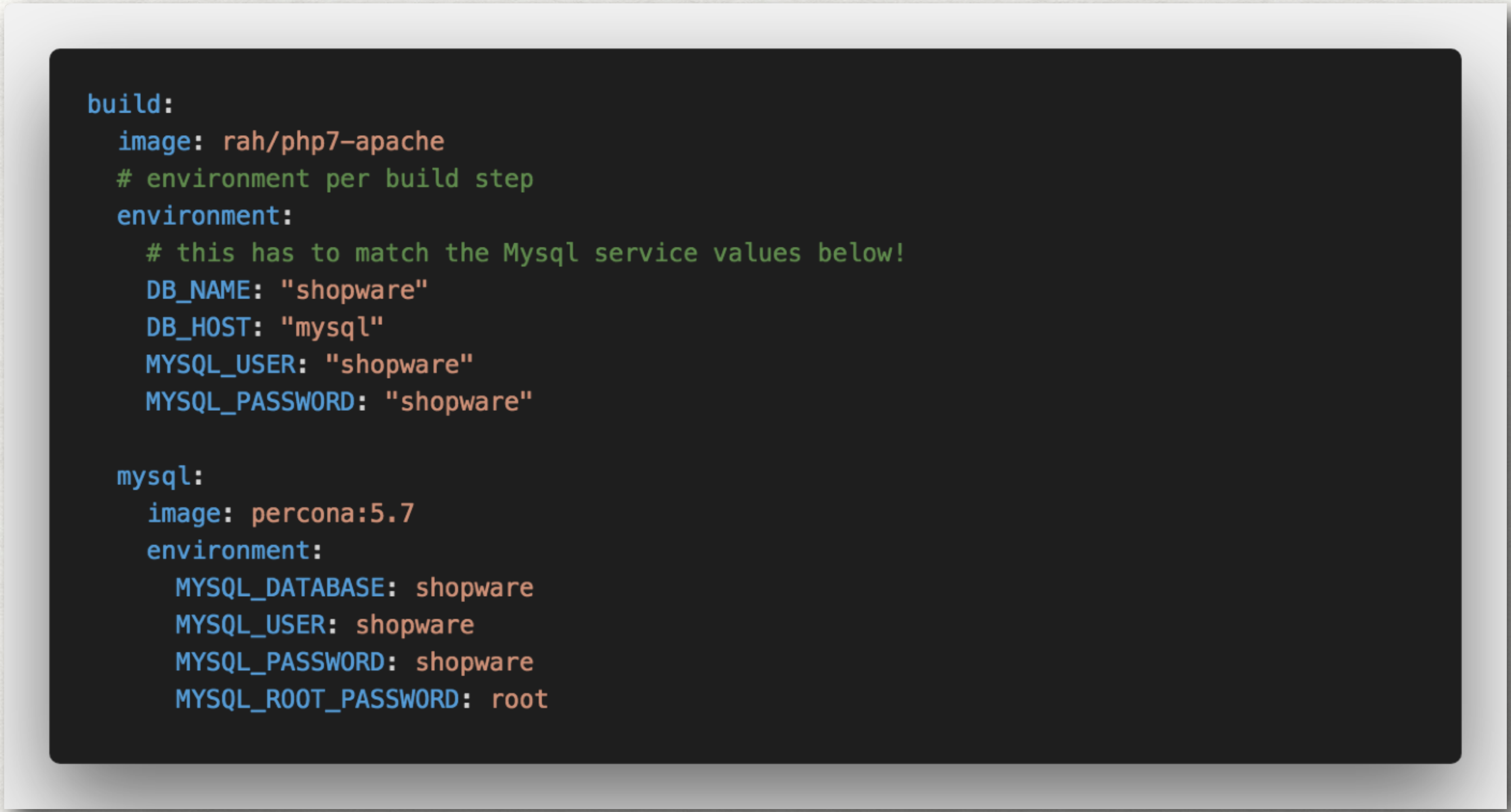


```
slack-notification:
    image: plugins/slack
    ...
    when:
      status: [ success, failure ]
      event: [ push, tag, deployment, pull_request ]
scp-deploy:
  when:
    environment: production
    event: deployment
    ...
matrix-build:
  when:
    matrix:
      GO_VERSION: 1.5
      REDIS_VERSION: 2.8
```

SERVICES

- allow you to run any container during the execution of your build process

- all services are in the same subnet with the process build containers

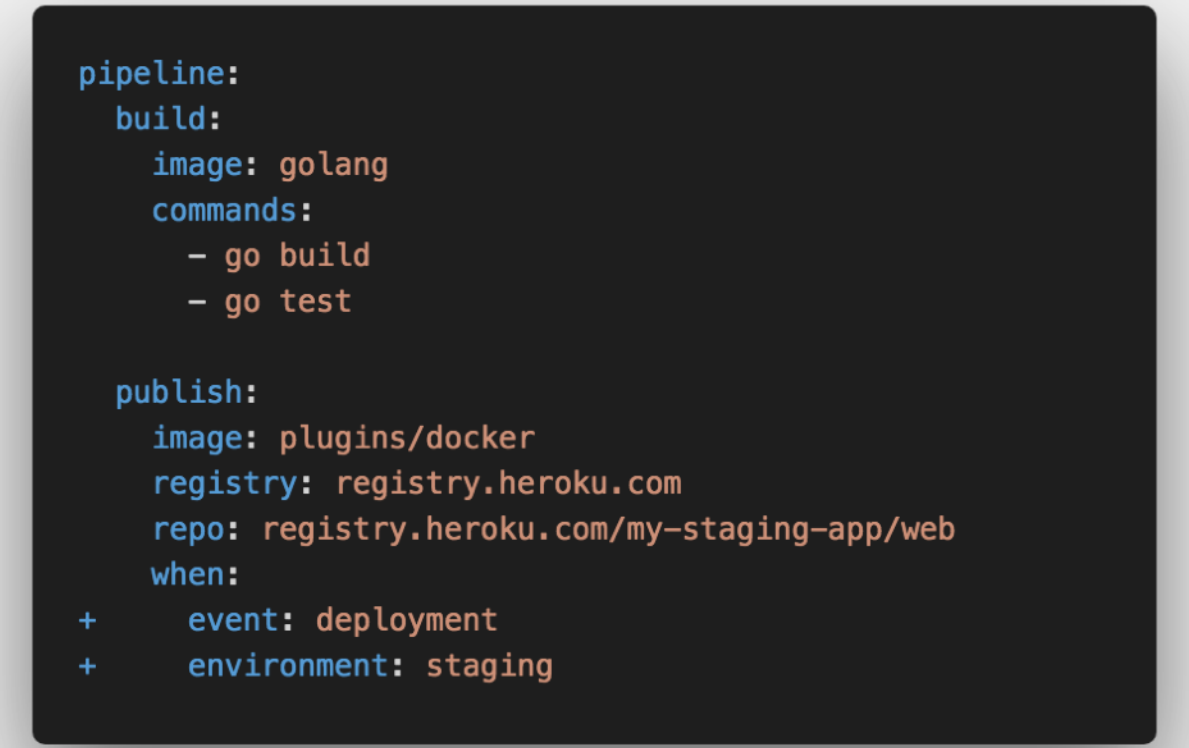


```yaml
build:
  image: rah/php7-apache
  # environment per build step
  environment:
    # this has to match the Mysql service values below!
    DB_NAME: "shopware"
    DB_HOST: "mysql"
    MYSQL_USER: "shopware"
    MYSQL_PASSWORD: "shopware"

  mysql:
    image: percona:5.7
    environment:
      MYSQL_DATABASE: shopware
      MYSQL_USER: shopware
      MYSQL_PASSWORD: shopware
      MYSQL_ROOT_PASSWORD: root
```

TRIGGER DEPLOYMENTS
("PROMOTE BUILDS")

- when you promote a commit or tag it triggers a new pipeline execution with event type deployment
- you can use the event type and target environment to limit step execution
- drone deploy <repo> <build> <environment>
- e.g. drone deploy octocat/hello-world 24 staging
- Not available via UI (see https://github.com/drone/drone-ui/pull/191) or API yet :(,
- but there are PRs / patches

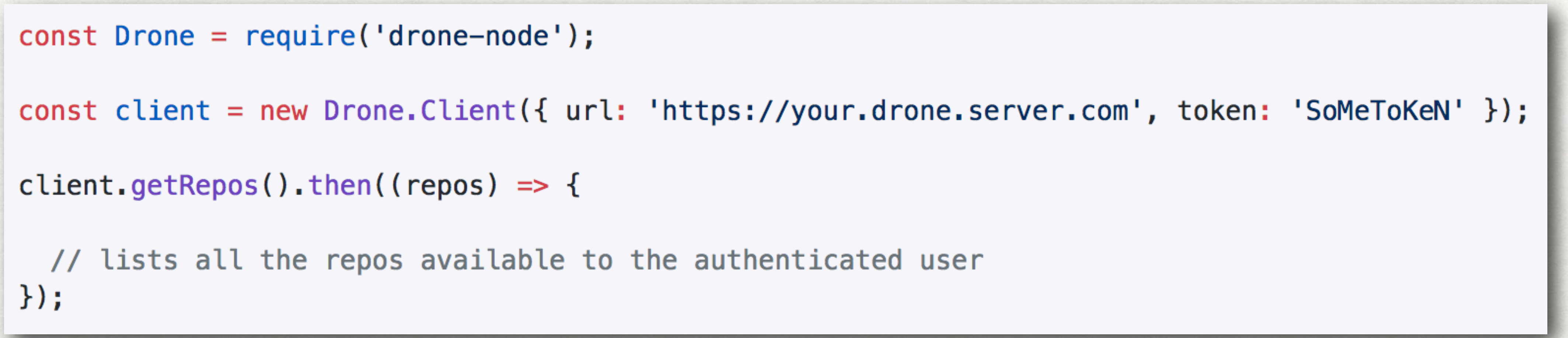
PROMOTE A BUILD



```
pipeline:
  build:
    image: golang
    commands:
      - go build
      - go test

  publish:
    image: plugins/docker
    registry: registry.heroku.com
    repo: registry.heroku.com/my-staging-app/web
    when:
+     event: deployment
+     environment: staging
```

APIS

- Drone offers a REST API with token authentication
- APIs in Node, Go, Python and Ruby, see http://docs.drone.io/api-overview/
- **Node API**, commands at https://github.com/drone/drone-node/blob/master/lib/client.js, example:

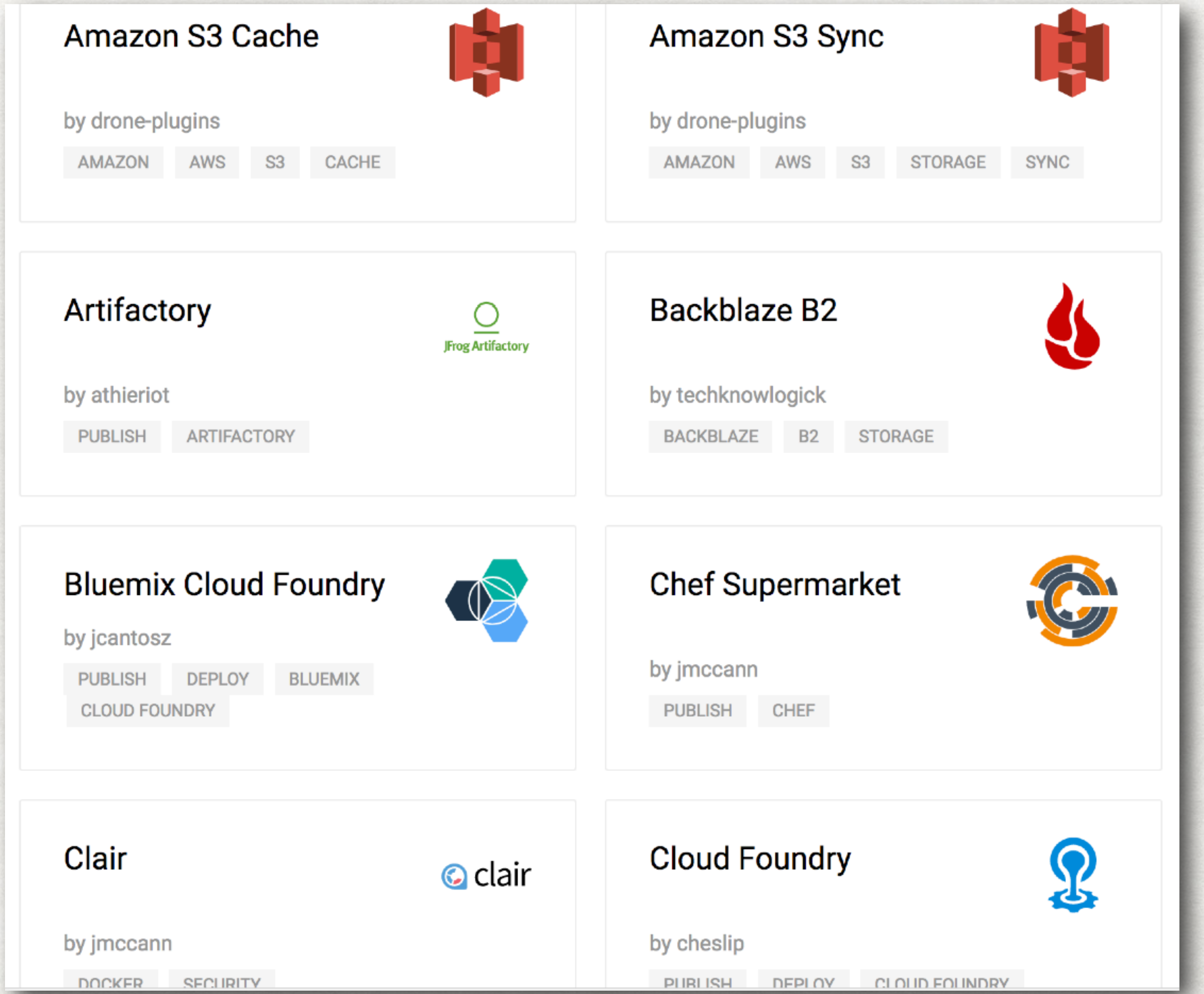


```
const Drone = require('drone-node');

const client = new Drone.Client({ url: 'https://your.drone.server.com', token: 'SoMeToKeN' });

client.getRepos().then((repos) => {

  // lists all the repos available to the authenticated user
});
```

PLUGINS

PLUGINS AS DOCKER IMAGES

- Plugins are Docker containers that perform pre-defined tasks and are configured as steps in your pipeline. Plugins can be used to deploy code, publish artifacts, send notification, and more
- Example: http://docs.drone.io/creating-custom-plugins-bash/



Amazon S3 Cache
by drone-plugins
AMAZON AWS S3 CACHE

Amazon S3 Sync
by drone-plugins
AMAZON AWS S3 STORAGE SYNC

Artifactory
by athieriot
PUBLISH ARTIFACTORY

Backblaze B2
by techknowlogick
BACKBLAZE B2 STORAGE

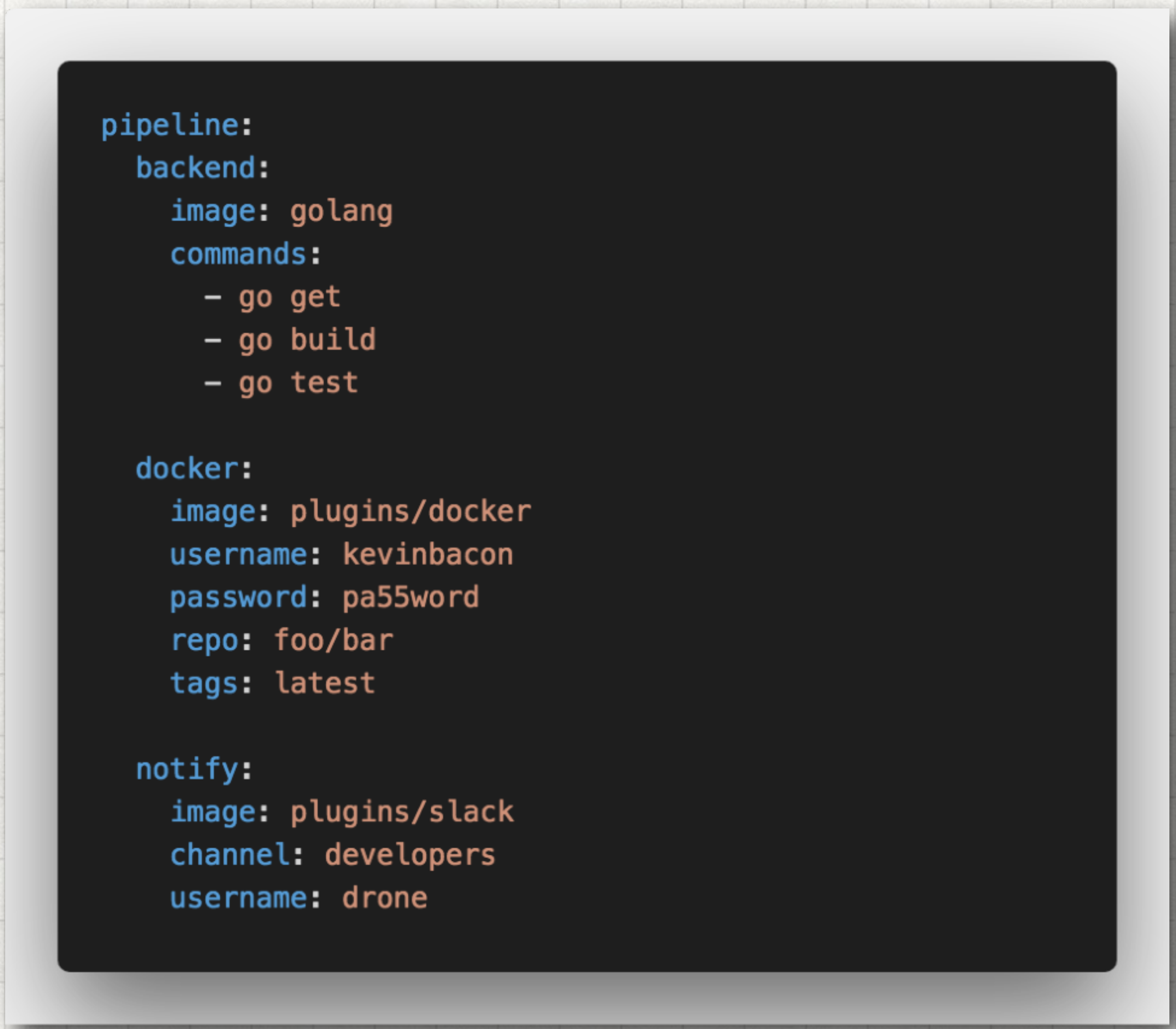
Bluemix Cloud Foundry
by jcantosz
PUBLISH DEPLOY BLUEMIX CLOUD FOUNDRY

Chef Supermarket
by jmccann
PUBLISH CHEF

Clair
by jmccann
DOCKER SECURITY

Cloud Foundry
by cheslip
PUBLISH DEPLOY CLOUD FOUNDRY

PLUG ME IN



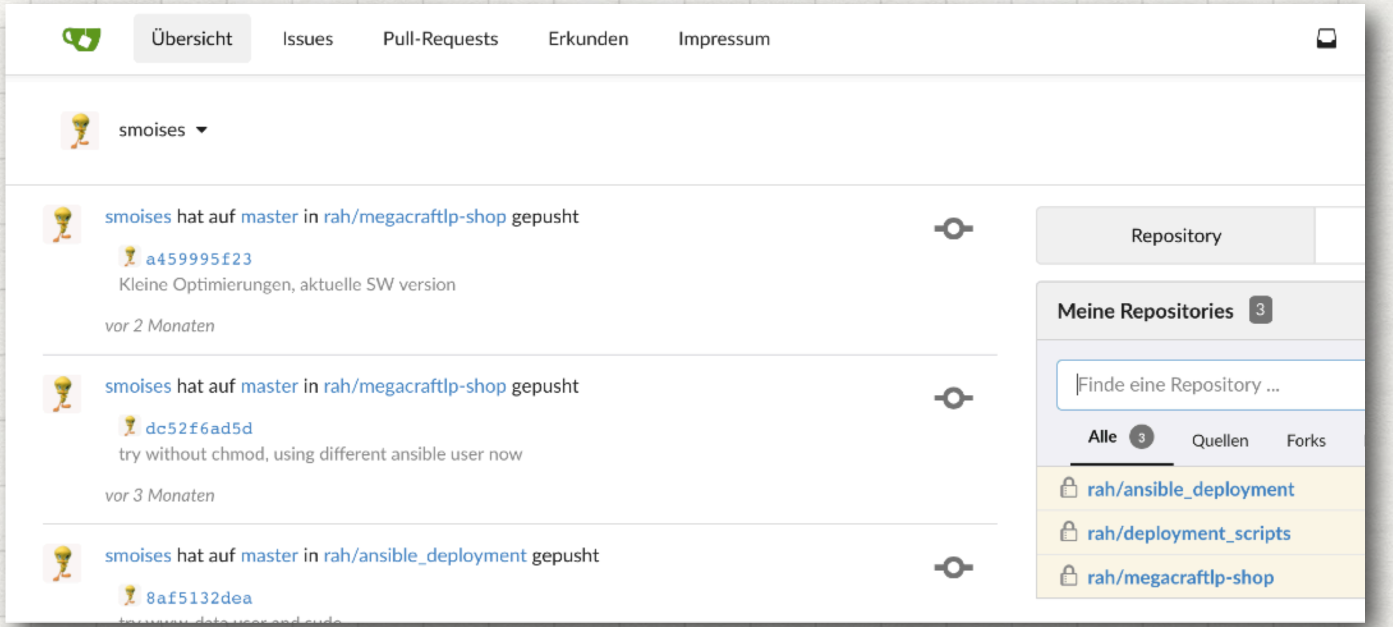
```
pipeline:
  backend:
    image: golang
    commands:
      - go get
      - go build
      - go test

  docker:
    image: plugins/docker
    username: kevinbacon
    password: pa55word
    repo: foo/bar
    tags: latest

  notify:
    image: plugins/slack
    channel: developers
    username: drone
```

DRONE LINKS

- https://drone.io/
- https://discourse.drone.io/ for support
- https://blog.maqpie.com/2017/03/21/build-and-deploy-applications-using-drone-ci-docker-and-ansible/
- https://drailing.net/2018/02/setting-up-continuous-delivery-with-drone/
- https://rancher.com/building-super-fast-docker-cicd-pipeline-rancher-droneci/

GITEA

GIT WITH A CUP OF TEA



Übersicht Issues Pull-Requests Erkunden Impressum

smoises

smoises hat auf master in rah/megacraftlp-shop gepusht
a459995f23
Kleine Optimierungen, aktuelle SW version
vor 2 Monaten

smoises hat auf master in rah/megacraftlp-shop gepusht
dc52f6ad5d
try without chmod, using different ansible user now
vor 3 Monaten

smoises hat auf master in rah/ansible_deployment gepusht
8af5132dea

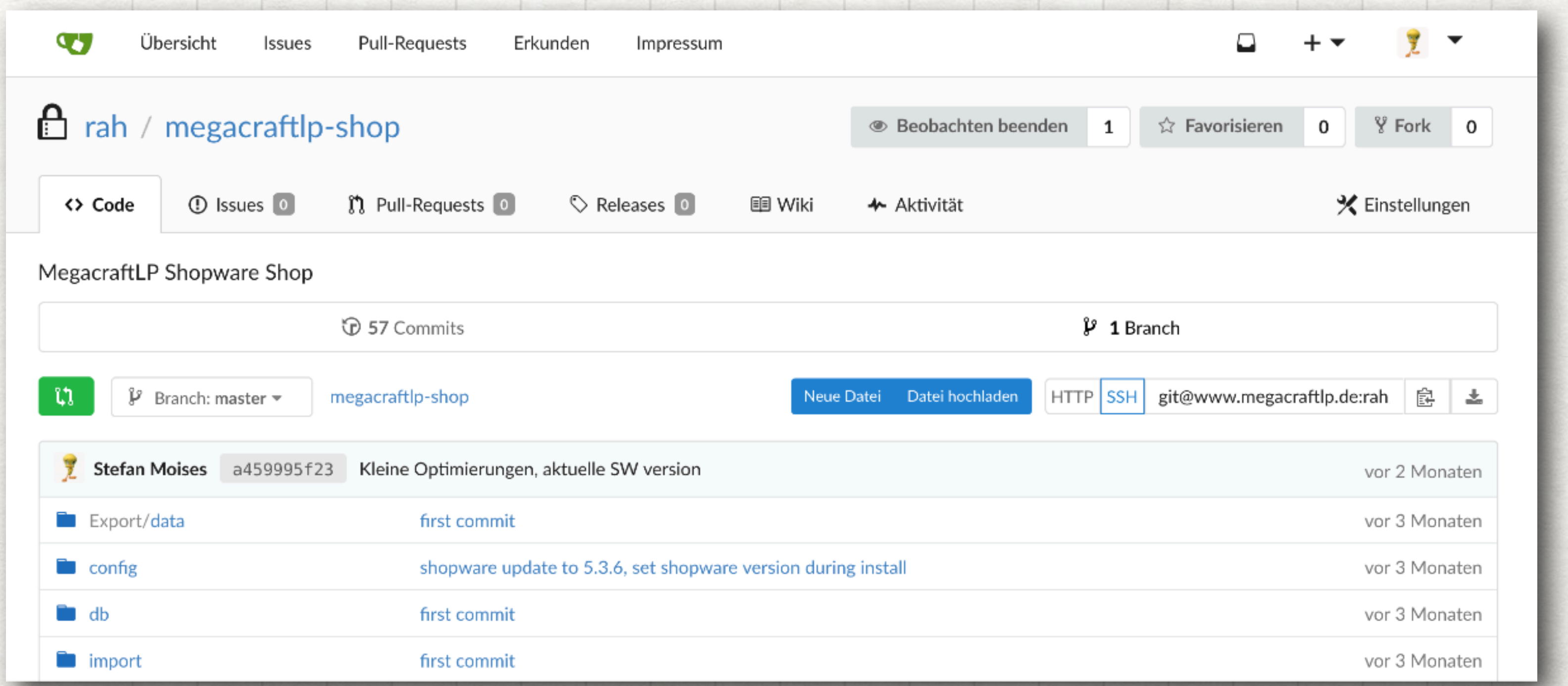
Repository

Meine Repositories 3

Finde eine Repository ...

Alle 3 Quellen Forks

- rah/ansible_deployment
- rah/deployment_scripts
- rah/megacraftlp-shop



Übersicht Issues Pull-Requests Erkunden Impressum

rah / megacraftlp-shop

Beobachten beenden 1 Favorisieren 0 Fork 0

Code | Issues 0 | Pull-Requests 0 | Releases 0 | Wiki | Aktivität | Einstellungen

MegacraftLP Shopware Shop

57 Commits 1 Branch

Branch: master megacraftlp-shop Neue Datei Datei hochladen HTTP SSH git@www.megacraftlp.de:rah

Stefan Moises a459995f23 Kleine Optimierungen, aktuelle SW version vor 2 Monaten

Export/data	first commit	vor 3 Monaten
config	shopware update to 5.3.6, set shopware version during install	vor 3 Monaten
db	first commit	vor 3 Monaten
import	first commit	vor 3 Monaten

GITEA

INSTALLATION

- via Docker
- from binary
- from source
- from package
- install e.g. using **systemd** on Ubuntu

```
sudo vim /etc/systemd/system/gitea.service
sudo systemctl enable gitea
sudo systemctl start gitea
```