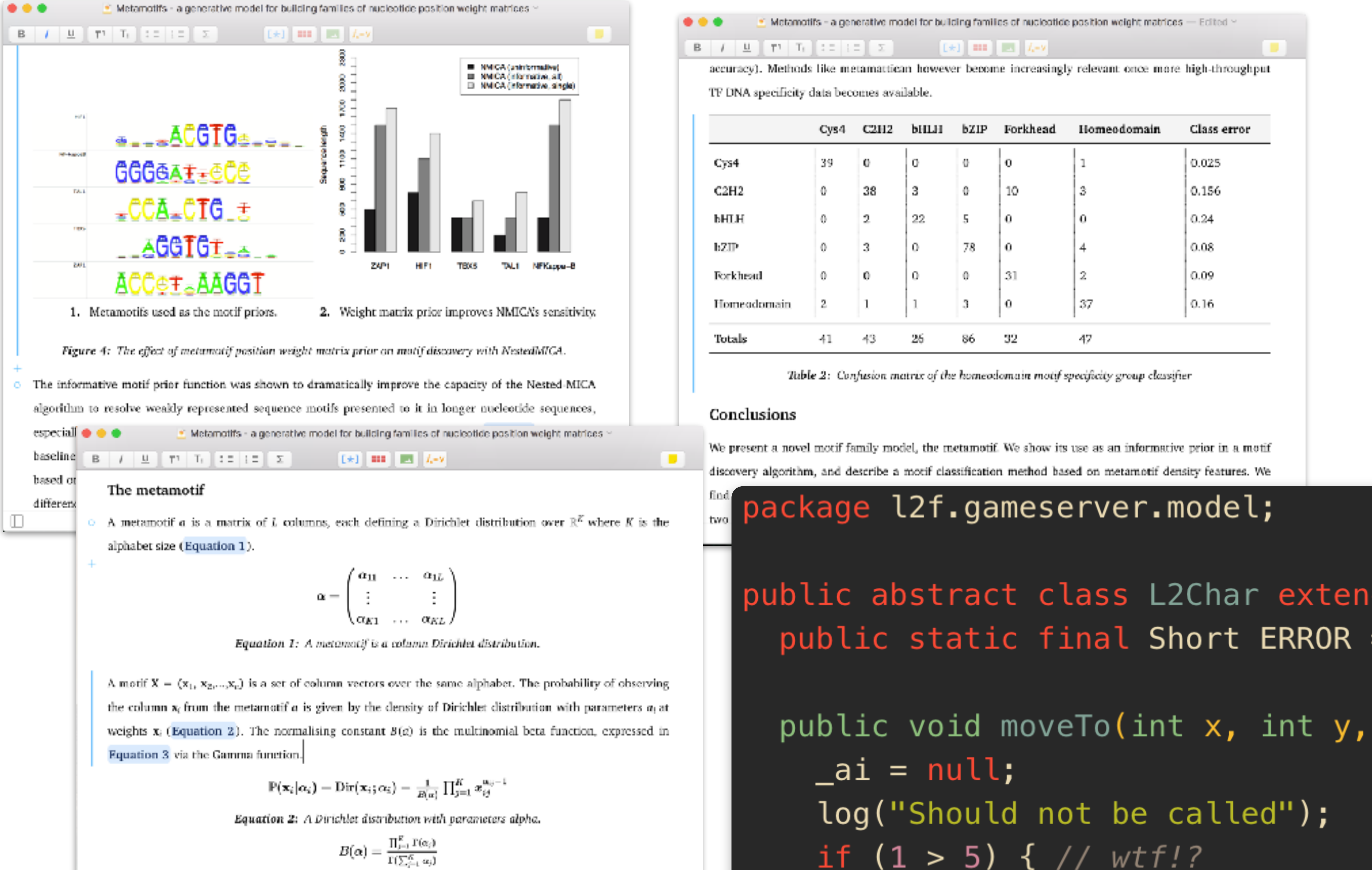
Dockerizing a Vapor app.

Matias Piipari (@mz2, manuscriptsapp.com)

Manuscripts in brief


Metamotifs - a generative model for building families of nucleotide position weight matrices
HIF1
NFKappaB
TAL1
TBX5
ZAP1
ACGTG
GGGGATTCCC
CCA CTG T
AGGTGT
ACCCT AAGGT
Sequence length
0
200
400
600
800
1000
1200
1400
1600
1800
2000
2200
NMICA (uninformative)
NMICA (informative, all)
NMICA (informative, single)
ZAP1
HIF1
TBX5
TAL1
NFKappa-B
Metamotifs - a generative model for building families of nucleotide position weight matrices — Edited
Metamotifs - a generative model for building families of nucleotide position weight matrices

1. Metamotifs used as the motif priors.

2. Weight matrix prior improves NMICA's sensitivity.

Figure 4: The effect of metamotif position weight matrix prior on motif discovery with NestedMICA.

The informative motif prior function was shown to dramatically improve the capacity of the Nested-MICA algorithm to resolve weakly represented sequence motifs presented to it in longer nucleotide sequences, especiall

baseline

based or

differen

accuracy). Methods like metamatician however become increasingly relevant once more high-throughput TF DNA specificity data becomes available.

	Cys4	C2H2	bHLH	bZIP	Forkhead	Homeodomain	Class error
Cys4	39	0	0	0	0	1	0.025
C2H2	0	38	3	0	10	3	0.156
bHLH	0	2	22	5	0	0	0.24
bZIP	0	3	0	78	0	4	0.08
Forkhead	0	0	0	0	31	2	0.09
Homeodomain	2	1	1	3	0	37	0.16
Totals	41	43	26	86	32	47	

Table 2: Confusion matrix of the homeodomain motif specificity group classifier

Conclusions

We present a novel motif family model, the metamotif. We show its use as an informative prior in a motif discovery algorithm, and describe a motif classification method based on metamotif density features. We

find

two

The metamotif

A metamotif α is a matrix of L columns, each defining a Dirichlet distribution over $\mathbb{R}^K$ where K is the alphabet size (Equation 1).

$$\alpha = \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1L} \\ \vdots & & \vdots \\ \alpha_{K1} & \cdots & \alpha_{KL} \end{pmatrix}$$

Equation 1: A metamotif is a column Dirichlet distribution.

A motif $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_L)$ is a set of column vectors over the same alphabet. The probability of observing the column $\mathbf{x}_i$ from the metamotif α is given by the density of Dirichlet distribution with parameters α_i at weights $\mathbf{x}_i$ (Equation 2). The normalising constant $B(\alpha)$ is the multinomial beta function, expressed in Equation 3 via the Gamma function.

$$\mathrm{P}(\mathbf{x}_i|\alpha_i) = \mathrm{Dir}(\mathbf{x}_i; \alpha_i) = \frac{1}{B(\alpha)} \prod_{j=1}^{K} x_{ij}^{\alpha_{ij}-1}$$

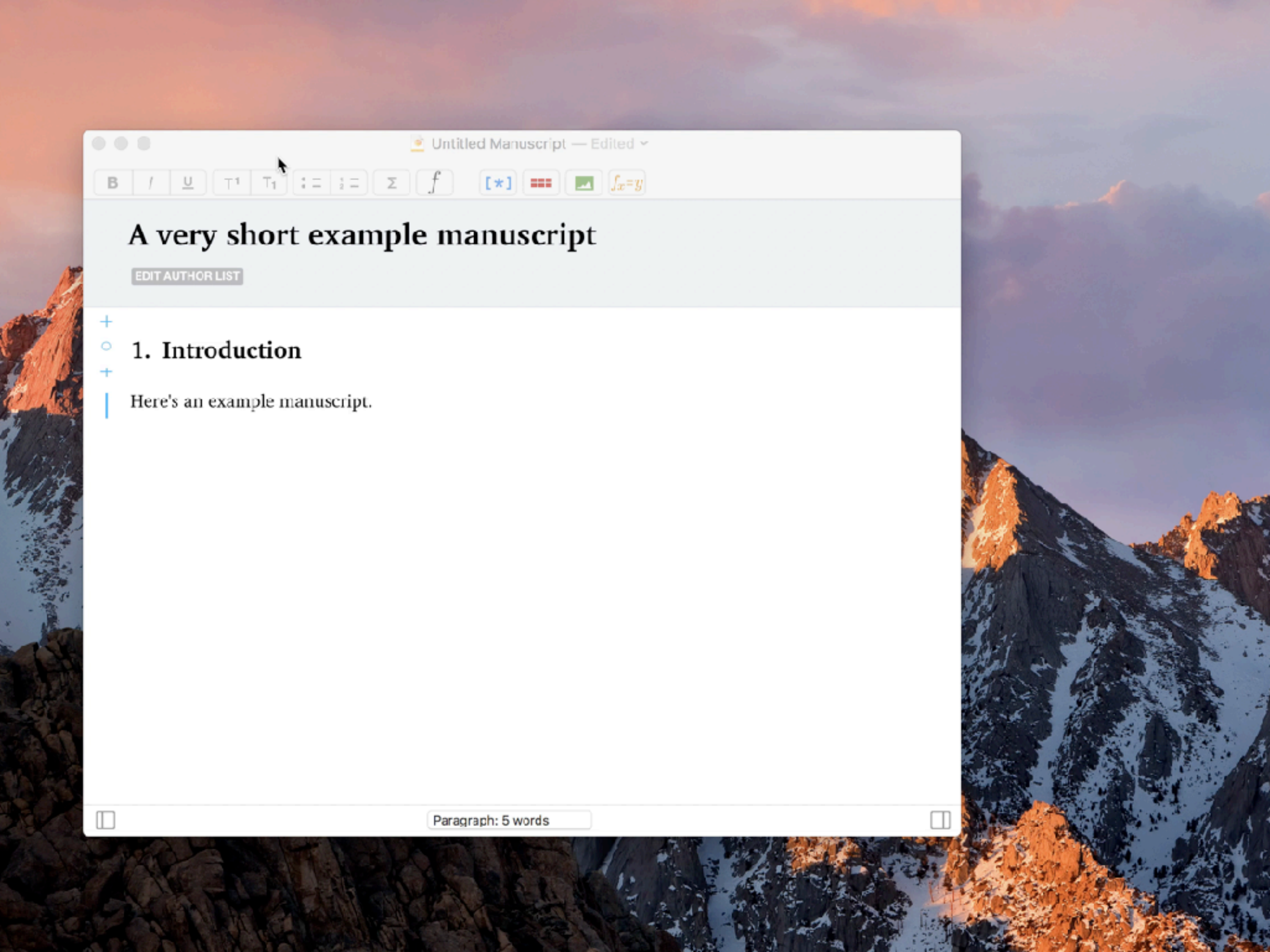
Equation 2: A Dirichlet distribution with parameters alpha.

$$B(\alpha) = \frac{\prod_{j=1}^{K} \Gamma(\alpha_j)}{\Gamma(\sum_{j=1}^{K} \alpha_j)}$$

```
package l2f.gameserver.model;

public abstract class L2Char exten
  public static final Short ERROR 

  public void moveTo(int x, int y,
    _ai = null;
    log("Should not be called");
    if (1 > 5) { // wtf!?
```



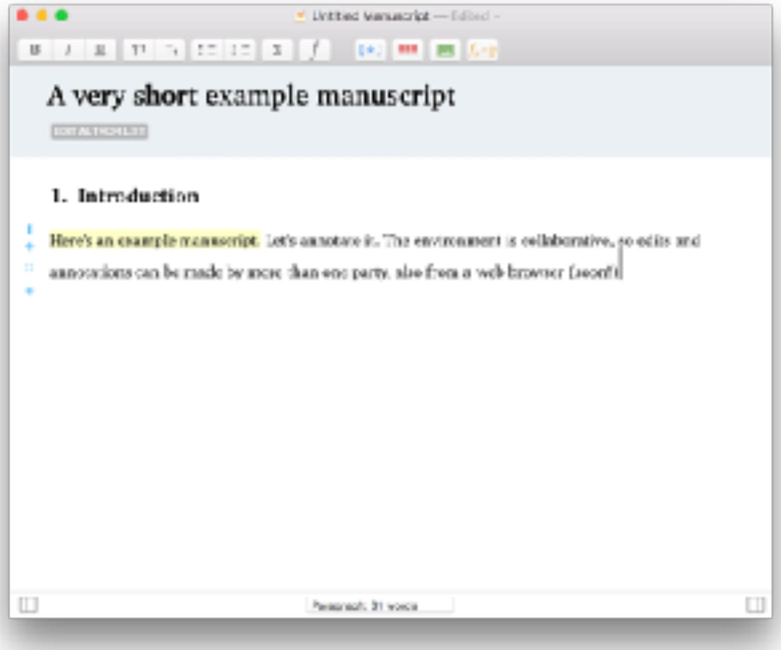
Untitled Manuscript — Edited

A very short example manuscript

1. Introduction

Here's an example manuscript.

Paragraph: 5 words



Untitled Manuscript — Edited

A very short example manuscript

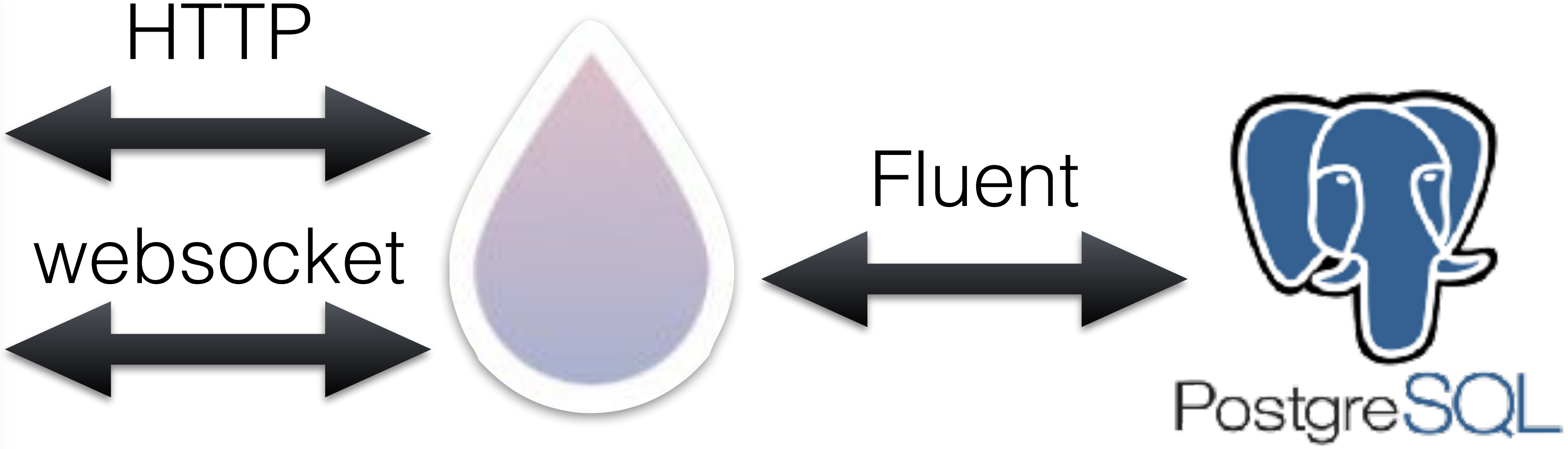
EDIT AUTHOR LIST

1. Introduction

Here's an example manuscript. Let's annotate it. The environment is collaborative, so edits and annotations can be made by more than one party, also from a web browser (soon!)

Paragraph, 31 words

Manuscripts for Mac



HTTP

websocket

Fluent

annotation data

Why Vapor?

- Fitting feature set: ORM, authentication with JWT, websockets.
- Cheap to run: high performance, low memory footprint.
- Code reuse nirvana awaits:
  share business logic between client and server.

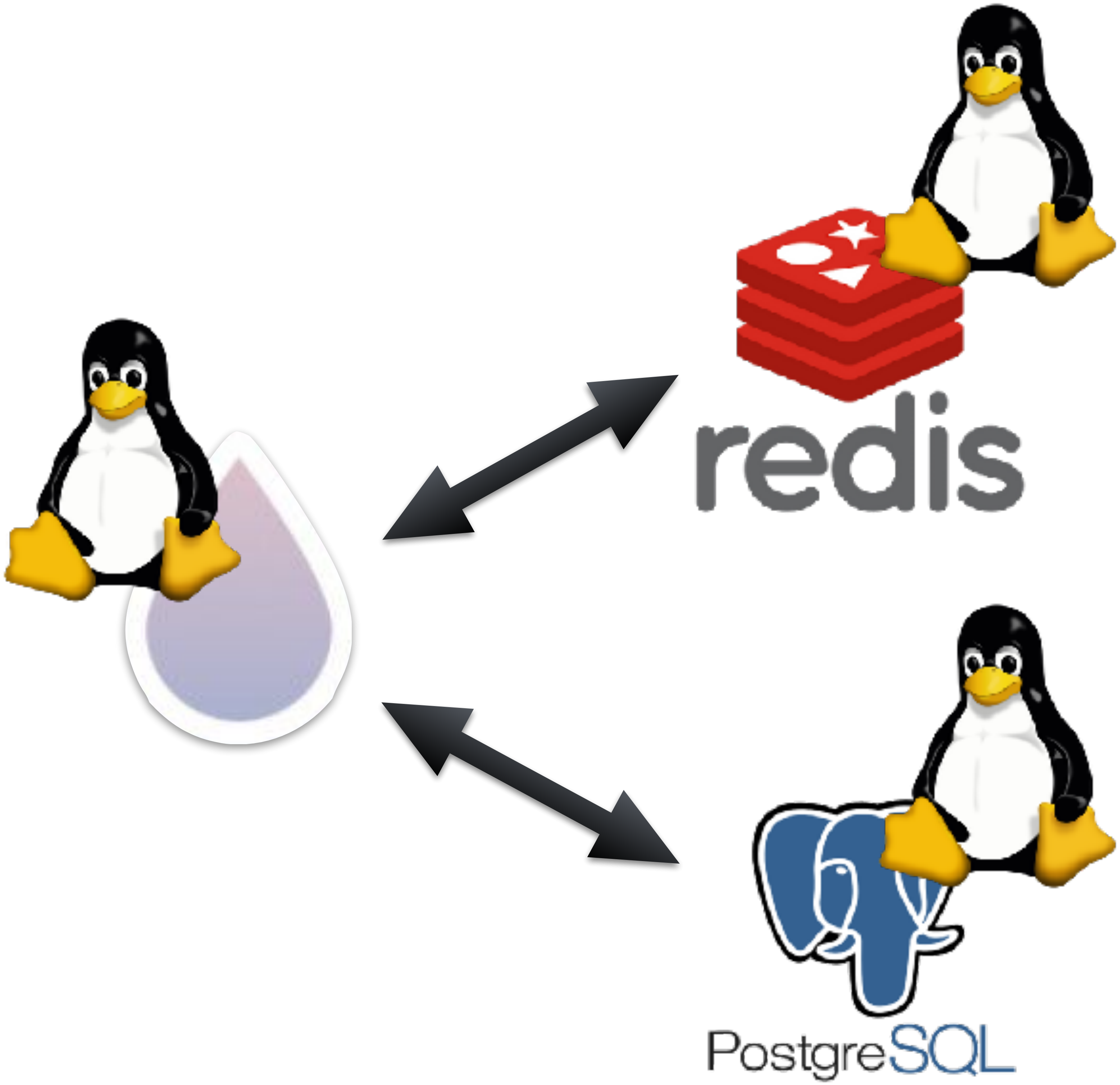
A new microservice is needed.

↓

An excuse to try Vapor!!!

🥳 🥳 🥳

Why put my app in a Docker container?


redis
PostgreSQL

Docker for development & debugging

- Often a cure to the “works on my machine” syndrome.
    - Repeatable, scripted configuration (Dockerfile) to run your application & dependencies (e.g. database).
    - Run Swift in a Linux environment similar to real thing.
    - Execute your tests in Linux using a Docker based CI runner.
- Start configuring image from a ready-made state (registry).
- You can spin up multiple containers too (docker-compose).
- Pretty fast and light: uses builtin macOS hypervisor.
(Linux as a host is faster still)

Docker for production

- Someone takes care of most crucial updates and the server lifecycle for you.
- Security by privilege separation.
  - Nothing exposed by default – you need to specify any ports, sockets, disk volumes to make visible through the host.

Blueprint of a Dockerfile

```
FROM ubuntu:14.04 # mandatory first line: the name of the base image.

# install package dependencies
RUN apt-get update
RUN apt-get install -y wget clang-3.6

# set guest environment variables
ENV PATH /usr/bin:$PATH

# copy current *host* working directory into guest path /vapor
ADD . /vapor

# set the *guest* working directory.
WORKDIR /vapor

# build your Vapor app
RUN swift build

# run your Vapor app
CMD .build/debug/App --env=production
```

Building and running a database server & app in docker

PostgreSQL

```
docker build -t postgresql github.com/sameersbn/docker-postgresql
docker run --name postgresql -itd --restart always

# You may want to script the post-launch admin tasks.
docker exec -it postgresql sudo -u postgres psql postgres \
-c "CREATE DATABASE \"…\";"
```

https://github.com/sameersbn/docker-postgresql

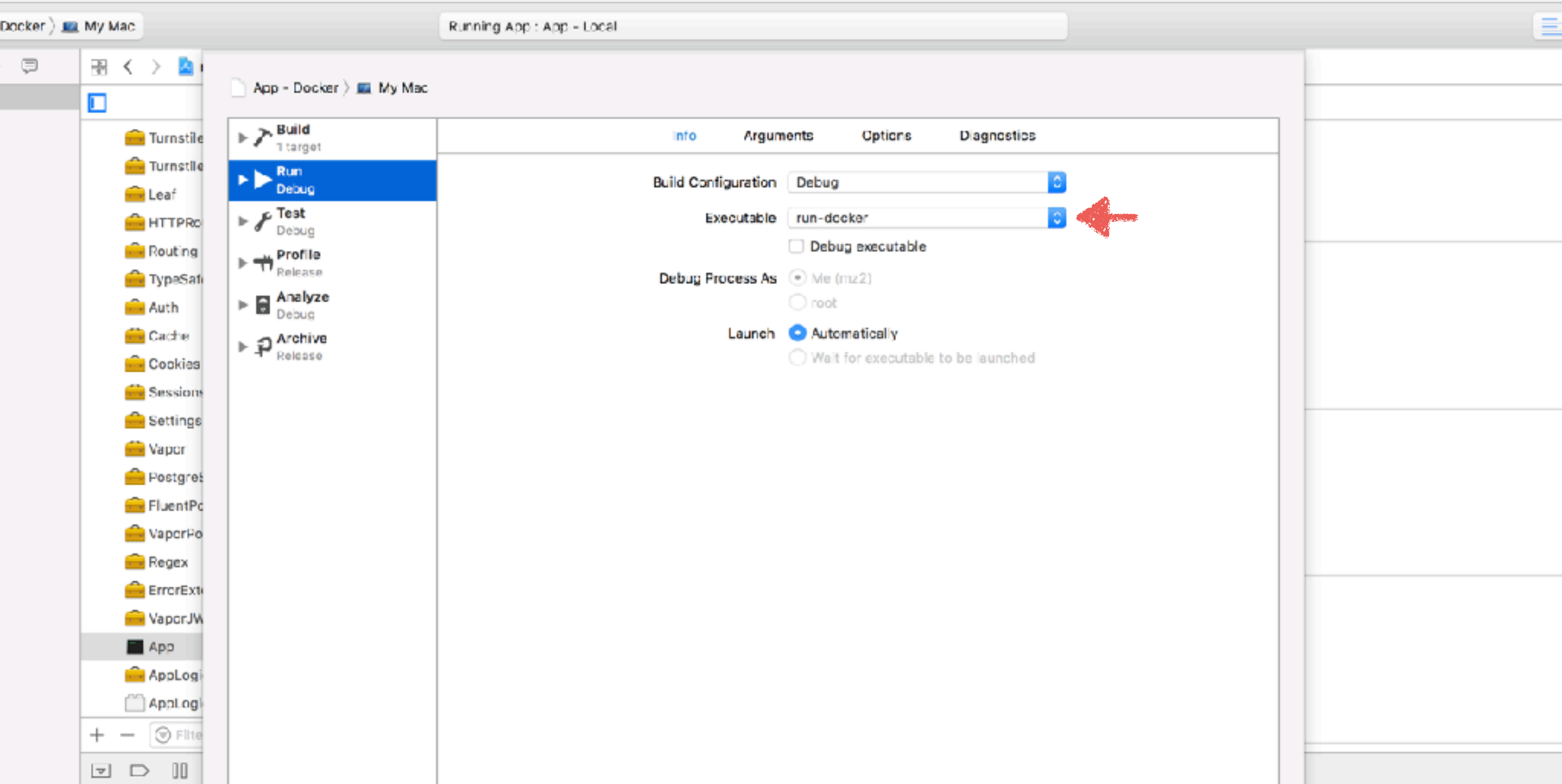
Your app

```
docker build -t manuscript-annotations -f Development/Dockerfile
docker run --rm -it -v $PWD:/vapor -p 5432 -p 8080:8080 manuscript-annotations
```

Dockerfile: https://gist.github.com/mz2/ae8b80fd06887d639a8992b988757e74

Running a docker app in Xcode

run-docker: https://gist.github.com/mz2/561dd8a7b0b7133c58062fc246799a4c
kill-docker: https://gist.github.com/mz2/0474b86c0261e415de422aaee4136e1c



Build Configuration: Debug

Executable: run-docker

☐ Debug executable

Debug Process As: ● Me (mz2) ○ root

Launch: ● Automatically ○ Wait for executable to be launched

Running tests in Docker & GitLab

1. Create a docker image to run tests with.

```
  docker login registry.gitlab.com
  docker build -f Tests/Dockerfile \
  -t registry.gitlab.com/mpapp-private/manuscript-annotations .
  docker push registry.gitlab.com/mpapp-private/manuscript-annotations
```

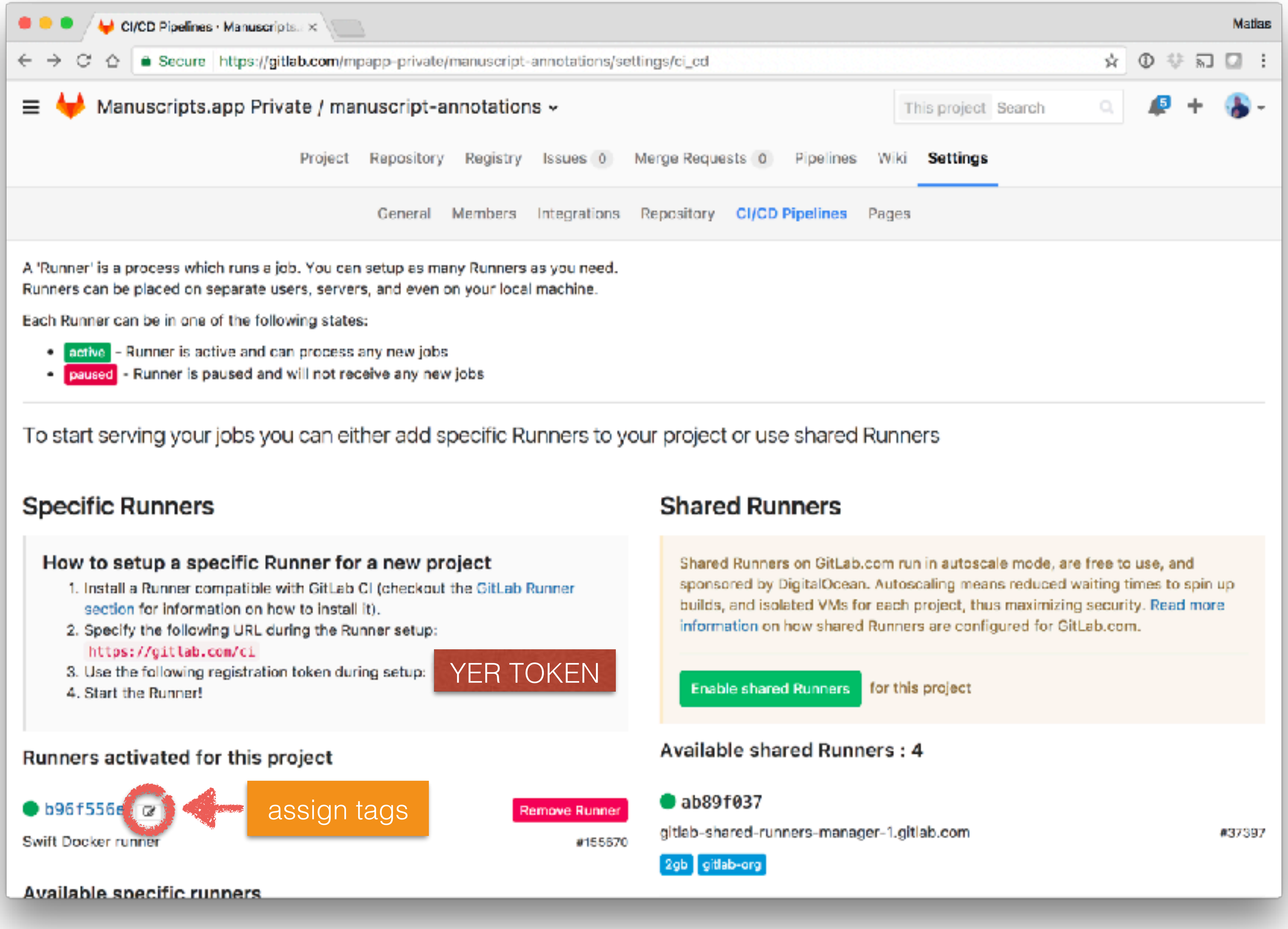
Tests/Dockerfile: https://gist.github.com/mz2/809fb9b3078b035bcd39bd27e240a613

2. Register a test runner.

```
gitlab-ci-multi-runner register \
--non-interactive \
--url https://gitlab.com/ci \
--registration-token “[PUT YER TOKEN HERE]” \
--description "Swift Docker runner" \
--executor "docker" \
--docker-image registry.gitlab.com/mpapp-private/manuscript-annotations:latest
```

3. Enable the runner.

Get registration token & enable the runner.



CI/CD Pipelines · Manuscripts...
Matias
Secure https://gitlab.com/mpapp-private/manuscript-annotations/settings/ci_cd
Manuscripts.app Private / manuscript-annotations
This project Search
Project
Repository
Registry
Issues 0
Merge Requests 0
Pipelines
Wiki
Settings
General
Members
Integrations
Repository
CI/CD Pipelines
Pages

A 'Runner' is a process which runs a job. You can setup as many Runners as you need. Runners can be placed on separate users, servers, and even on your local machine.

Each Runner can be in one of the following states:

- active - Runner is active and can process any new jobs
- paused - Runner is paused and will not receive any new jobs

To start serving your jobs you can either add specific Runners to your project or use shared Runners

Specific Runners

How to setup a specific Runner for a new project

1. Install a Runner compatible with GitLab CI (checkout the GitLab Runner section for information on how to install it).
2. Specify the following URL during the Runner setup: `https://gitlab.com/ci`
3. Use the following registration token during setup: YER TOKEN
4. Start the Runner!

Runners activated for this project

● b96f556e

Swift Docker runner

#155670

Available specific runners

Shared Runners

Shared Runners on GitLab.com run in autoscale mode, are free to use, and sponsored by DigitalOcean. Autoscaling means reduced waiting times to spin up builds, and isolated VMs for each project, thus maximizing security. Read more information on how shared Runners are configured for GitLab.com.

for this project

Available shared Runners : 4

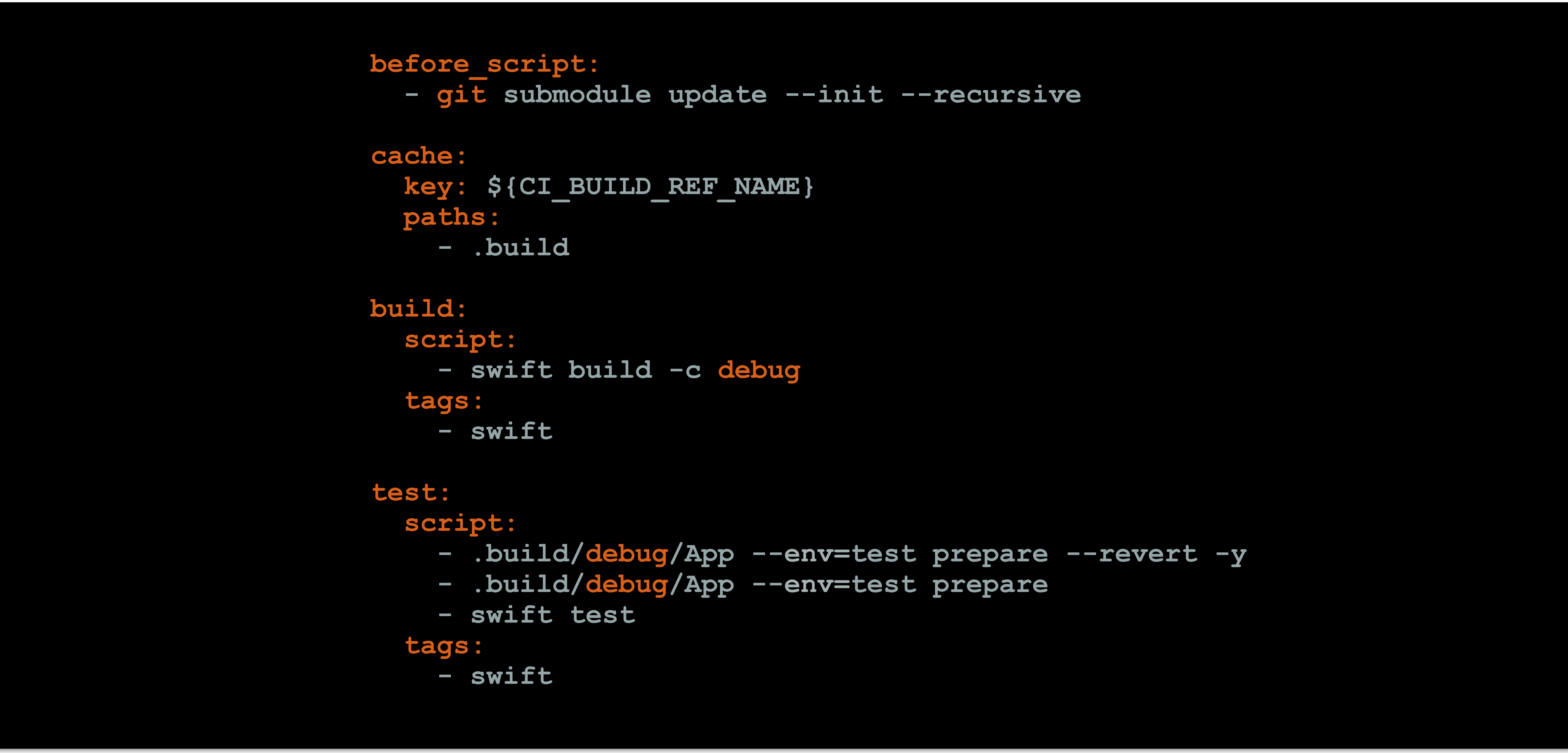
● ab89f037

gitlab-shared-runners-manager-1.gitlab.com #37397

2gb gitlab-org

Running tests in Docker & GitLab

An example .gitlab-ci.yml



```
before_script:
  - git submodule update --init --recursive

cache:
  key: ${CI_BUILD_REF_NAME}
  paths:
    - .build

build:
  script:
    - swift build -c debug
  tags:
    - swift

test:
  script:
    - .build/debug/App --env=test prepare --revert -y
    - .build/debug/App --env=test prepare
    - swift test
  tags:
    - swift
```

.gitlab-ci.yml: https://gist.github.com/mz2/65c32e691ae55d1b8c3e49391fdd986a

Deploying to Heroku with Docker

```
heroku plugins:install heroku-container-registry
heroku container:login
heroku container:push
```

Dockerfile: https://gist.github.com/mz2/a70694d7f260b46013055bf8b1380e9e

More info:
https://devcenter.heroku.com/articles/container-registry-and-runtime

Why deploy to Heroku with Docker?

- Use what you learned when building your development env.
- Avoid lock-in to Heroku (lots of Docker based hosts around).
- Deployed image preparation happens on your host => no 15min build timeouts.
- Subjective: creating a Dockerfile for running an arbitrary thing on Heroku is easier than scripting a buildpack.

Differences between development, test, production Dockerfile configuration.

Development:

```
WORKDIR /vapor
EXPOSE 8080

# mount in local sources via:  -v $(PWD):/vapor
VOLUME /vapor

CMD swift build && .build/debug/App --env=development
```

Production:

```
ADD . /vapor
WORKDIR /vapor
RUN swift build
CMD echo "PORT: ${PORT}" && .build/debug/App --env=production
```

Test:

(no VOLUME, EXPOSE, ADD or CMD needed – CI runner does it all.)