First steps for Startups

Continuous Integration as Code

By Ing. Gary Ascuy

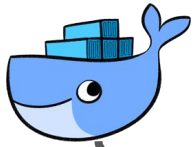
About Me - Ing. Gary Ascuy Anturiano

Mail: gary.ascuy@gmail.com / gary.ascuy@jalasoft.com
Mobile: +591 727 596 93 (BO) / +57 320 4431 029 (CO)

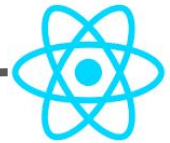
- Senior Software Developer at **Jala Colombia SAS**
- Software Architect as Freelance / Free Time
- I like **Robotics**/Electronics/**Front-End**/Back-End/DevOps/...
- Volunteer/Co-Organizer at **Docker Cochabamba**

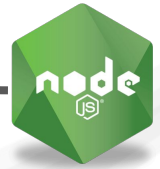
- github.com/gary-ascuy
- medium.com/@gary.ascuy
- linkedin.com/in/gary-ascuy-6619bbb9

Agenda



- Startups - Design Sprint
- Continuous { Integration / Delivery / Deployment }
- Docker Ultra Fast Overview
- GitLab CI & Runners (Shared / Specific)
- Configure Frontend (React Project)
- Configure Backend (NodeJS API)





Startups - Facts / Objective

MONEY

Startups - Facts / Objective

TIME

=

MONEY

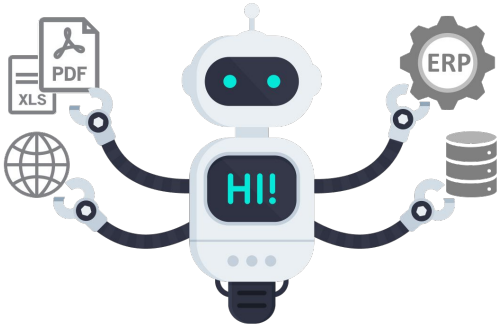
Startups - Facts / Wrong Objectives

Startups - What we need to do?

- **Reduce**(task time) ... but **HOW?**

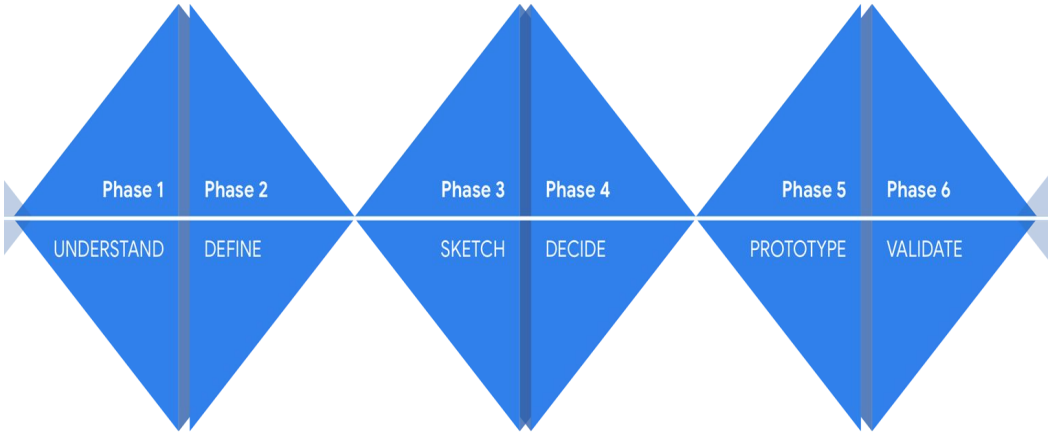
Design Sprint
Validate Ideas

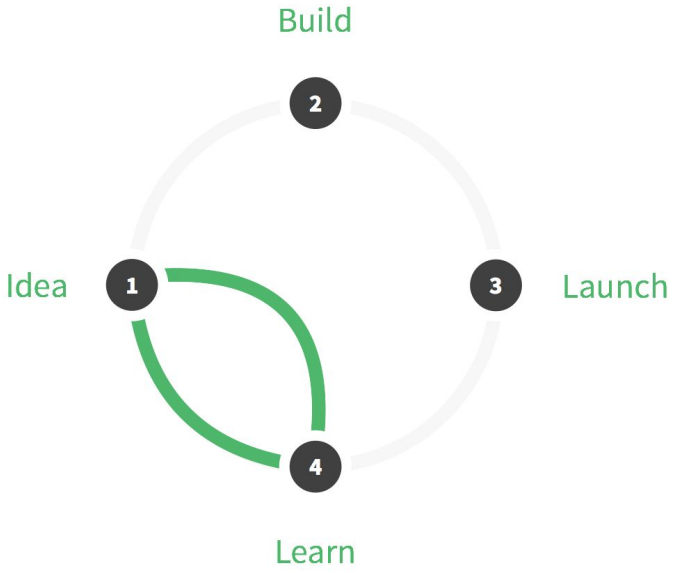
&



Automation
CI / CD

Startups - Design Sprint





Startups - Design Sprint / Multidisciplinary Team



- Developers
- Automation
- Quality Engineer ...

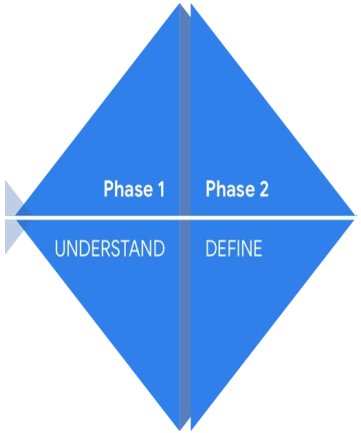


- Managers
- Stakeholders
- Scrum Master ...



- Designers
- Marketing
- Researcher / Experts ...

Startups - Design Sprint / Methods



Phase 1: Understand
User Journey Mapping

Phase 1: Understand
Experience Mapping

Phase 1: Understand
HMW Voting

Phase 1: Understand
User Interviews

PHASE 1
UNDERSTAND

Phase 2: Define
Success Metrics & Signals

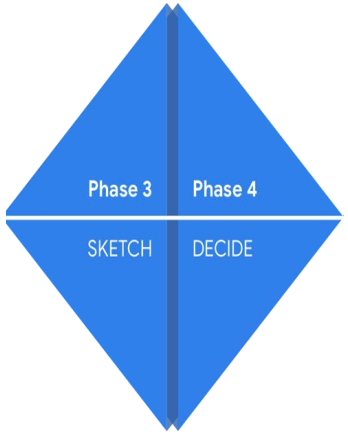
Phase 2: Define
Design Principles

Phase 2: Define
The Golden Path

Phase 2: Define
Future Press Release

PHASE 2
DEFINE

Startups - Design Sprint / Methods



Phase 3: Sketch
Boot Up Note Taking

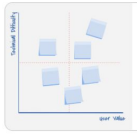
Phase 3: Sketch
Crazy 8's

Phase 3: Sketch
Crazy 8's Sharing and Voting

Phase 3: Sketch
Solution Sketch

Phase 4: Decide
Present Solution Sketches

Phase 4: Decide
Dot Vote



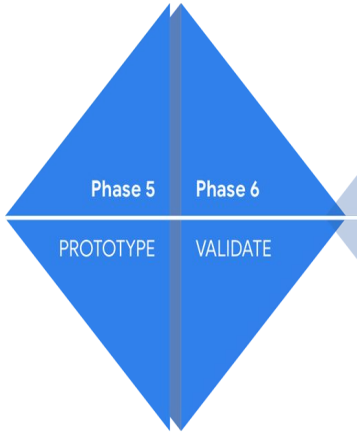
Phase 4: Decide
Decision Matrix

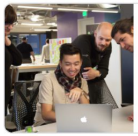
Phase 4: Decide
Rumble or All-In-One

PHASE 3
SKETCH

PHASE 4
DECIDE

Startups - Design Sprint / Methods

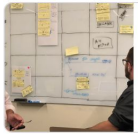




Phase 5: Prototype
Prototyping Tools

Phase 5: Prototype
Tips for Prototyping Stage

Phase 5: Prototype
Create a Kanban Board

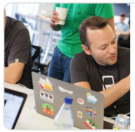


Phase 5: Prototype
Narrate the Storyboard

Phase 6: Validate
Usability Study

Phase 6: Validate
Cognitive Walkthroughs

Phase 6: Validate
Stakeholder Review



Phase 6: Validate
Technical Review

PHASE 5
PROTOTYPE

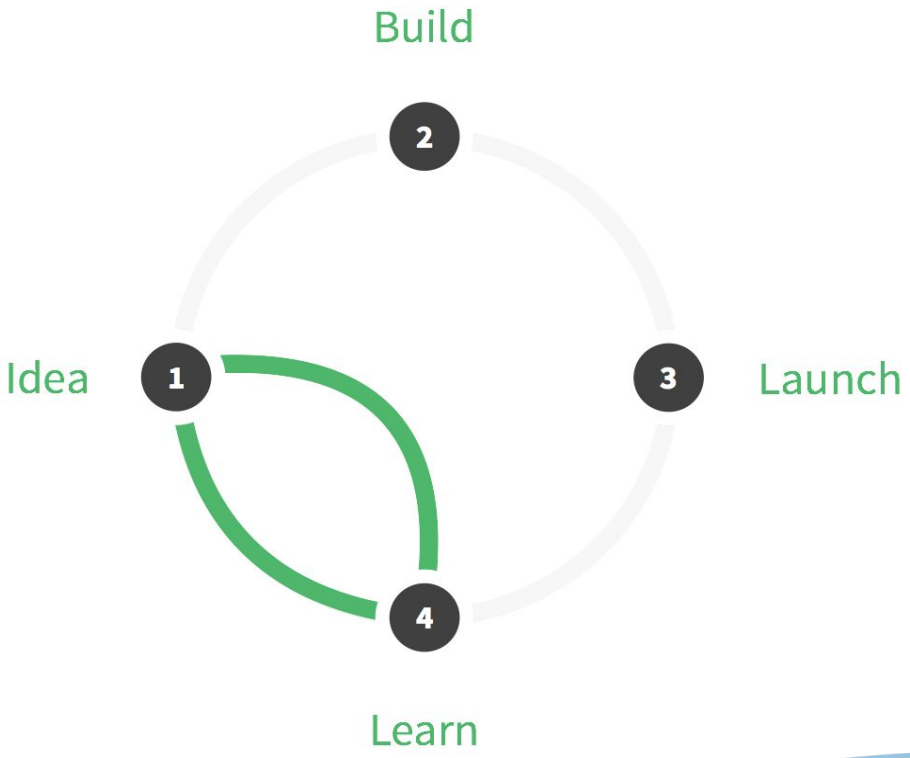
PHASE 6
VALIDATE

Startups - Design Sprint

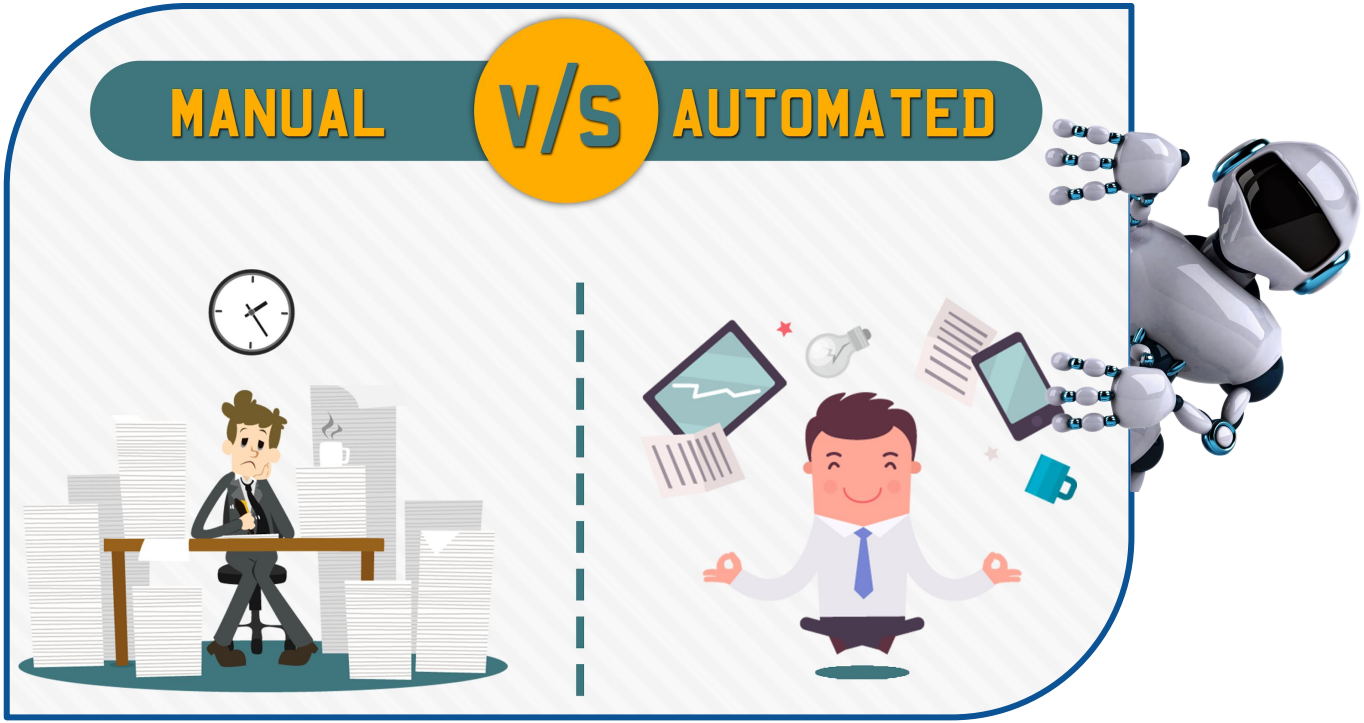
The **important thing** is that you **pick the methods** that **work best for your specific goal** and plan the number of days for your Sprint accordingly.

Learn, explore, create and find out **what works best** for the types of problems you typically seek to solve.

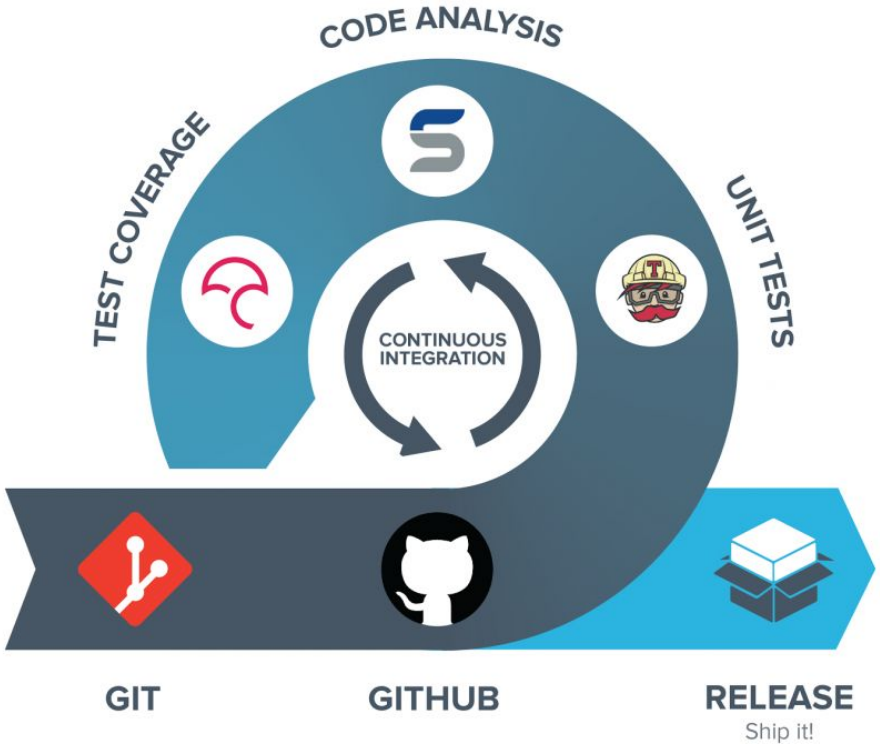
Startups - Design Sprint



Continuous { Integration / Delivery / Deployment }



Continuous { Integration / Delivery / Deployment }



Continuous { Integration / Delivery / Deployment }

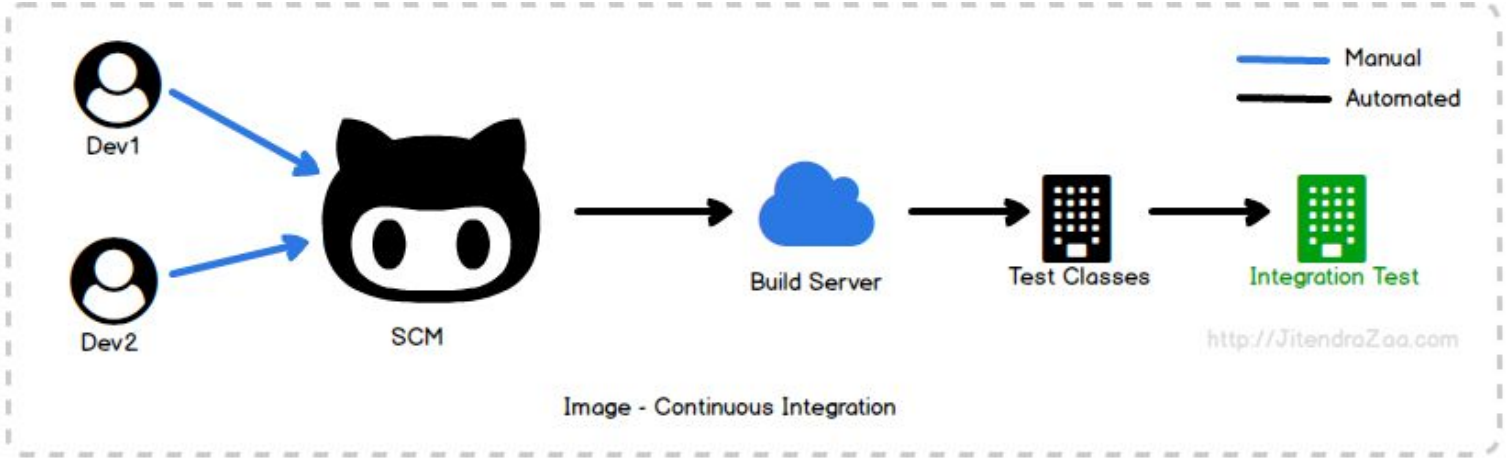


Image - Continuous Integration

Continuous { Integration / Delivery / Deployment }

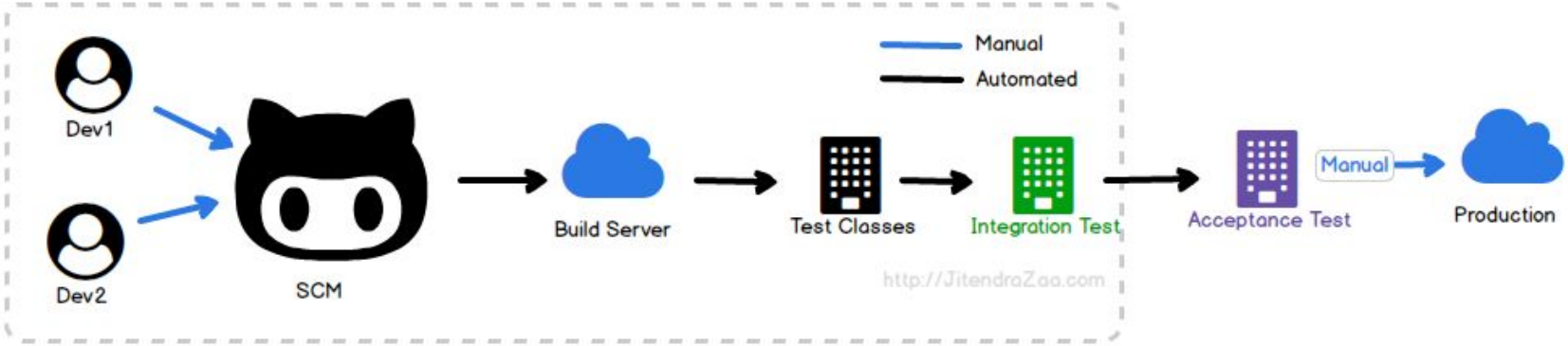


Image - Continuous Delivery

Continuous { Integration / Delivery / Deployment }

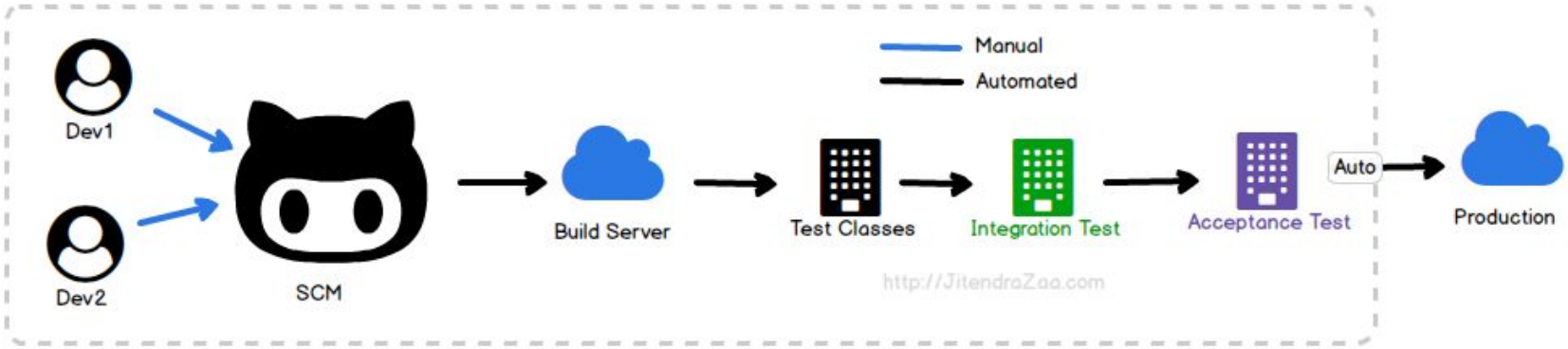


Image - Continuous Deployment

Continuous { Integration / Delivery / Deployment }

DEVELOPERS

PUSH

GITLAB CI

SOURCE CODE

PUBLISH

TEST

STAGING

PRODUCTION

Continuous { Integration / Delivery / Deployment }

TeamCity

cloudbees

urban {code}

BuildMaster

Octopus Deploy

AWS CodePipeline

SEMAPHORE

AppVeyor

Buddy

Travis CI

Bamboo

DRONE

Buildkite

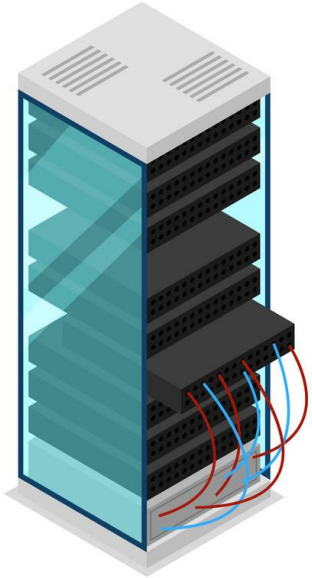
circleci

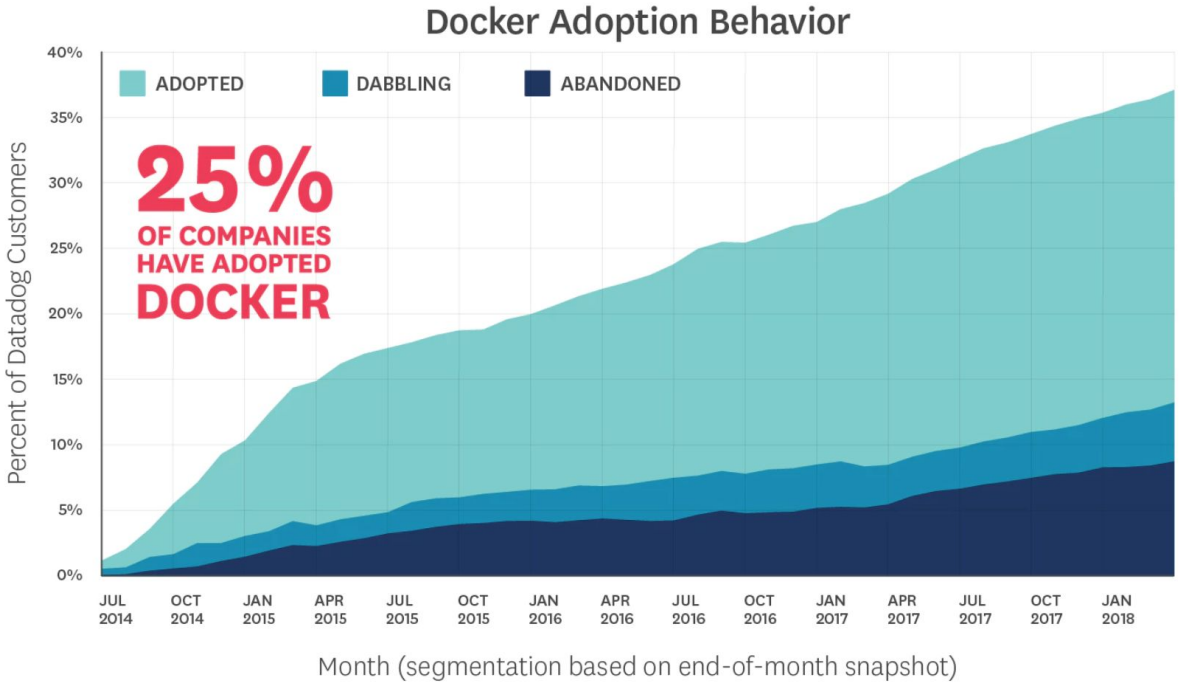
go

GitLab

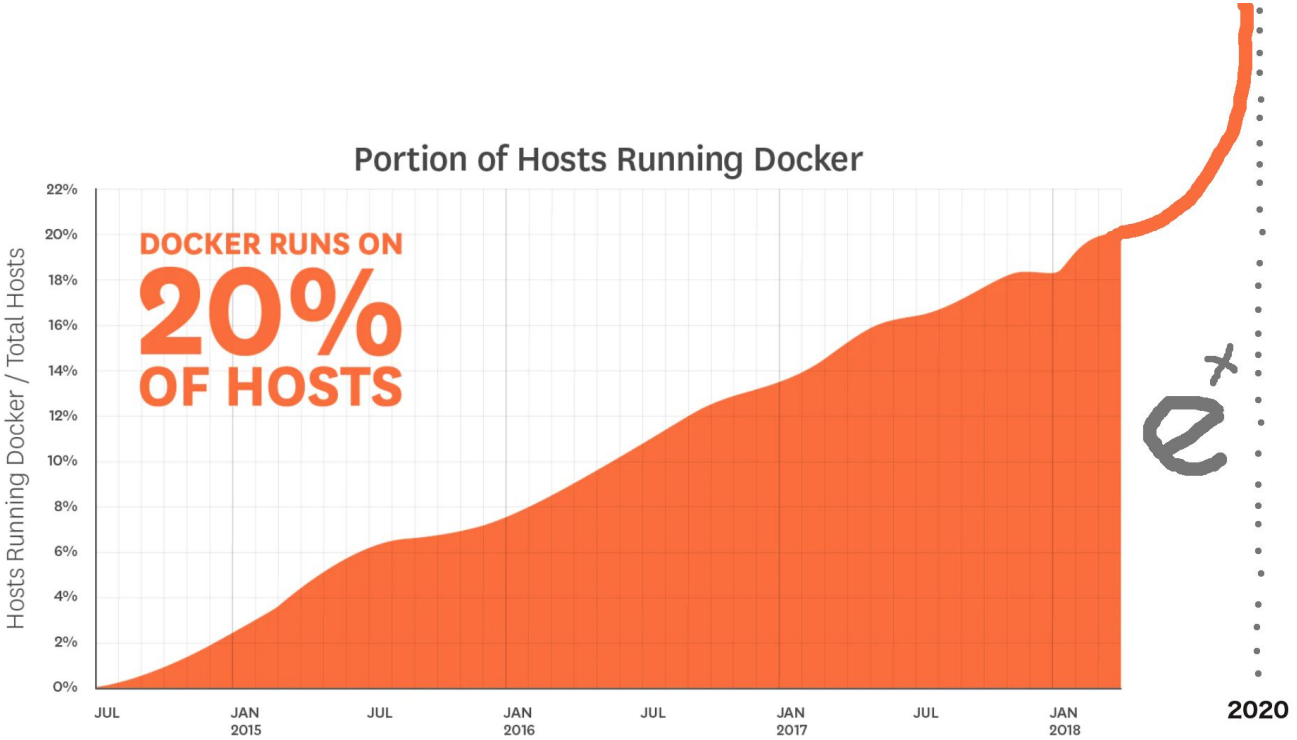
Docker Ultra Fast Overview / Adoption (WHY?)

PRODUCTION





Docker Ultra Fast Overview / Expectation



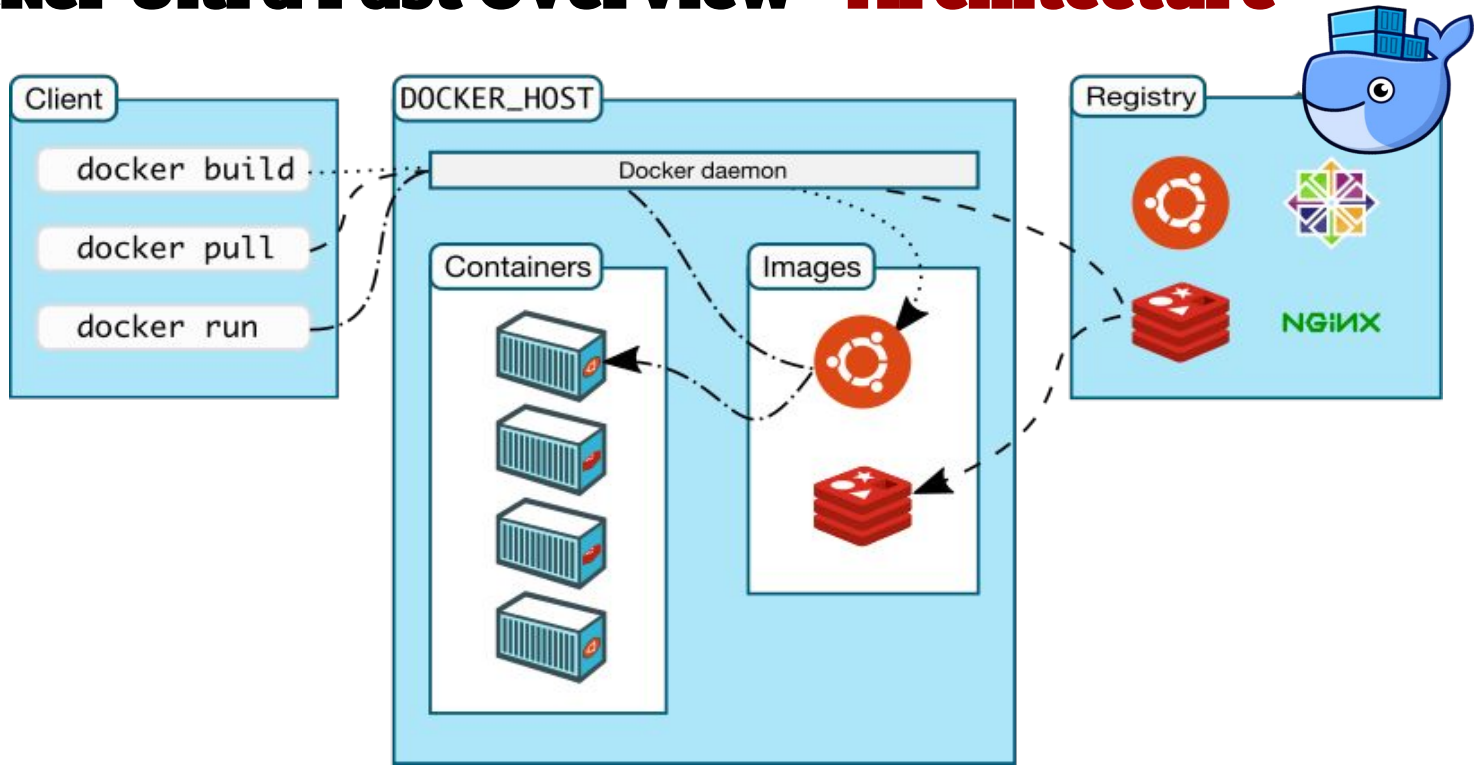
Docker Ultra Fast Overview - What is?

3 2 1

*Securely **build**, **share** and **run** modern applications anywhere*

-- Docker Developer

Docker Ultra Fast Overview - Architecture

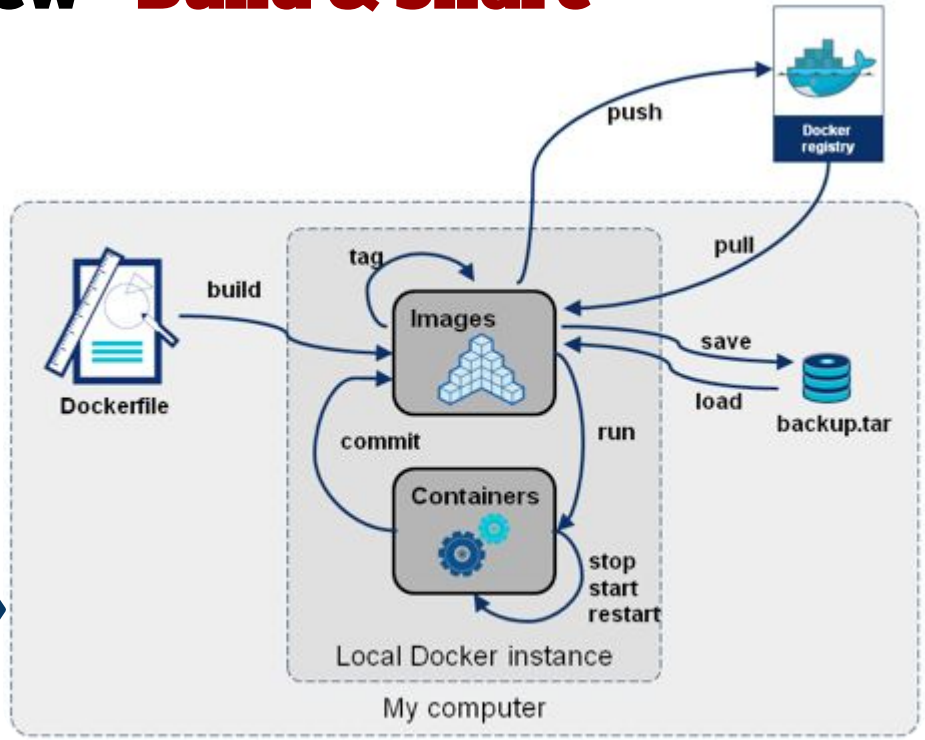


Docker Ultra Fast Overview - Run

```
$ docker run -p 80:80 nginx

$ docker run -p 27017:27017 mongo

$ docker run -p 27018:27017 mongo:3.6

$ docker run -p 3306:3306 mysql

$ docker run -p 5666:5672 -p 15666:15672 \
                    rabbitmq:3.7-management
```

Docker Ultra Fast Overview - Build & Share

```
# Dockerfile
FROM node:10.4-alpine
LABEL maintainer="gary.ascuy@gmail.com"

ENV NODE_ENV=production
RUN apk add --no-cache imagemagick
COPY ./build ./build
RUN npm install

CMD node build/app.js
```



GitLab CI / CD - BASE CODE

GitLab CI / CD - Architecture

CI/CD

GITLAB SERVER

1 GitLab process

RUNNER SERVER

0..n Runner process

YOUR LAPTOP

0..n Runner process

GitLab CI / CD - Get Started

DEVELOPERS

PUSH

GitLabCi

GITLAB CI
SOURCE CODE

GitLab Runner

GitLab Runner

GitLab Runner

Developper Runner

container

container

container

GITLAB - RUNNERS
AGENTS

GitLab CI / CD - Get Started

DEVELOPERS

PUSH

.gitlab-ci.yml

```
test:
  image: node:latest
  script:
    - echo 'Hello World - CI as Code'
```

GitLab CI / CD - Run Test Jobs - Back / Front

DEVELOPERS

PUSH

.gitlab-ci.yml

```
test api:
 image: node:latest
 before_script:
   - cd api
   - yarn install
 script:
   - yarn lint
   - yarn test

test ui:
 image: node:latest
 before_script:
   - cd ui
   - yarn install
 script:
   - yarn lint
   - yarn test
```

GitLab CI / CD - Run Test Jobs - REFACTOR

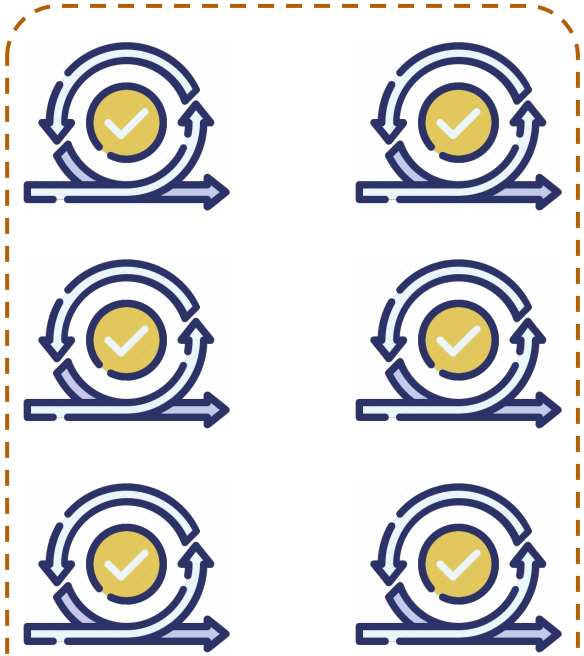
DEVELOPERS

PUSH

.gitlab-ci.yml

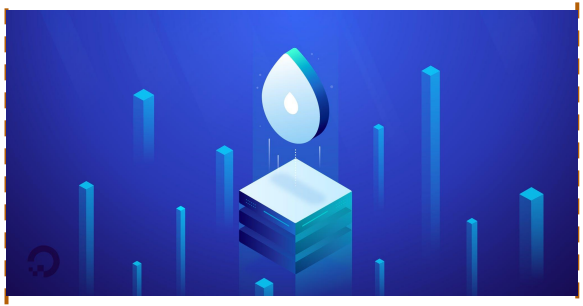
```
.test:
  image: node:latest
  before_script:
    - cd $PROJECT_PATH
    - yarn install
  script:
    - yarn lint
    - yarn test

test api:
  extends: .test
  variables: { PROJECT_PATH: api }

test ui:
  extends: .test
  variables: { PROJECT_PATH: ui }
```

GitLab CI & Runners (Shared / Specific)



SHARED RUNNERS

DigitalOcean



SPECIFIC RUNNERS

GitLab CI & Runners (Shared / Specific)

```
$ export TOKEN=<TOKEN FROM GITLAB>
$ docker run --rm -it -v $(pwd):/etc/gitlab-runner \
    gitlab/gitlab-runner:latest register \
    --non-interactive --name ciAsCode \
    --url https://gitlab.com --registration-token "$TOKEN" \
    --executor docker --docker-image docker:latest \
    --docker-volumes "/var/run/docker.sock:/var/run/docker.sock"
    --docker-volumes "/cache"
```

```
$ docker run -d --name gitlab-runner --restart always \
    -v $(pwd):/etc/gitlab-runner \
    -v /var/run/docker.sock:/var/run/docker.sock \
    gitlab/gitlab-runner:latest
```

GitLab CI / CD - Run Test Jobs - BUILD / PUSH

DEVELOPERS

PUSH

.gitlab-ci.yml

```
stages:
  - test
  - build
  - deploy

test api: ...
test ui: ...

build:
  image: docker/compose:latest
  stage: build
  services:
    - docker:dind
  before_script:
    - echo "$PASS" | docker login -u ...
  script:
    - docker-compose build
    - docker-compose push
  after_script:
    - echo 'Done'
```

GitLab CI / CD - Run Test Jobs - DEPLOY

DEVELOPERS

PUSH

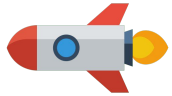
.gitlab-ci.yml

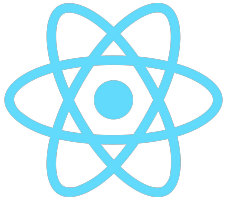
```
stages:
  - test
  - build
  - deploy

test api: ...
test ui: ...

build: ...

deploy:
  image: docker/compose:latest
  stage: deploy
  services:
    - docker:dind
  before_script:
    - echo "$PASS" | docker login -u ...
  script:
    - docker-compose down
    - docker-compose up -d
```

Conclusion A - Optimize($f(t)$)

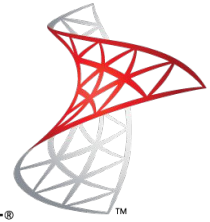
$$\text{MIN}\{f(\text{⏳})\}$$

Conclusion B - Smart Decisions

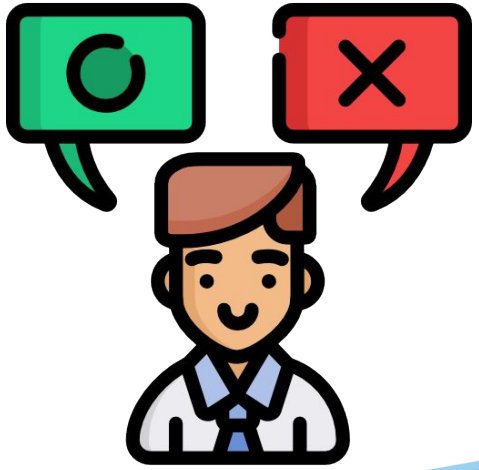


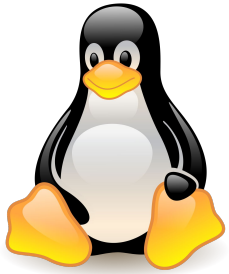


mongoDB



Microsoft® SQL Server®






QUESTIONS


Thank You!

References

- **Design Sprint**
  - https://designsprintkit.withgoogle.com/methodology/overview
- **Docker**
  - https://docs.docker.com/get-started/overview/
  - https://docs.docker.com/compose/
- **GitLab CI**
  - https://docs.gitlab.com/ee/ci/
  - https://docs.gitlab.com/ee/ci/introduction/#continuous-integration (short and direct)
  - https://docs.gitlab.com/ee/ci/introduction/#continuous-delivery (short and direct)
  - https://docs.gitlab.com/ee/ci/introduction/#continuous-deployment (short and direct)