DevOps for Defense

February 2020

Life of a Microservice @Shipt

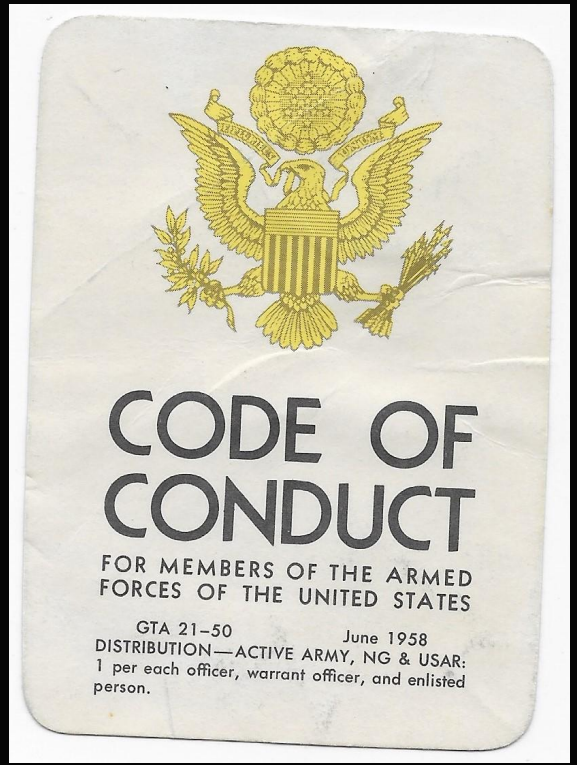
Vadim Uchitel

Sponsored by: TEKsystems Own change

https://devopsfordefense.org
https://www.meetup.com/DevOps-for-Defense/
https://github.com/jondavid-black/DevOpsForDefense
devopsfordefense@gmail.com
https://twitter.com/devops4defense

DevOps for Defense Meetup: Code of Conduct

- UNCLASSIFIED ONLY!!!!
- Treat each other with respect and professionalism.
- Do not talk about private, sensitive, or proprietary work.
- Do talk about your experiences, needs, desires to improve work in our domain.
- Do share your thoughts.
- Do learn from others.
- Do respect & tip your bartenders!



What's Next for DevOps for Defense?

March 5th

Hans Dockter
CEO Gradle Inc.
Powerful Automation & Insight

April 2nd

Open source complete CI/CD toolchain out-of-the-box.

Beyond

(Still coordinating & planning, but here's what we're thinking.)

Nicolas M. Chaillan
USAF Chief Software Officer -
Bringing DevSecOps DoD-wide

Dr. Mik Kersten
CEO Tasktop -
Author of Project to Product

Hack-a-thon?
Opportunity to put our DevOps learning into practice.

Provide us feedback so we can tailor to your needs.

DevOps for Defense Community Challenge

Donate Time 1 Day a Month

https://girlswhocode.com/

Emeka Barclay Marshall
Language Arts | Liberty Middle Scho
Apple Teacher
Google Certified Educator
Microsoft Innovative Educator
Flipgrid Certified Educator

"Knowledge is power. Information is liberating. Education is the premise of progress, in every society, in every family." -Kofi Annan

Website: caffeinatedteacher.weebly.com
Twitter: @teacheremeka

Help Drive Positive Change

Support Girls Who Code through smiles.amazon.com

Consider Subscribing & Supporting

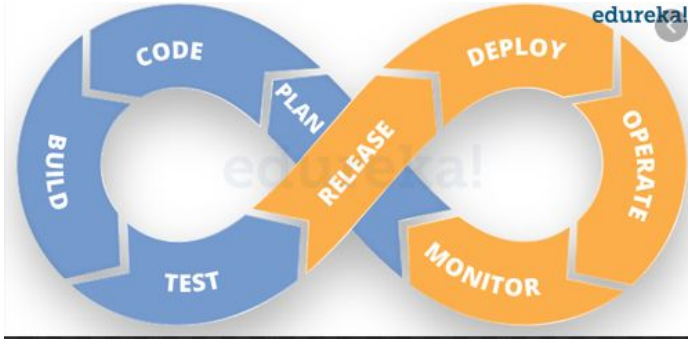
Subscribe to
DevOps for Defense
on **YouTube**

Consider supporting us through GitHub Sponsors.
Multiple levels of support. 100% Match!

https://github.com/jondavid-black
No fees! - All donations directly support the future of DevOps for Defense meetup!

Life of a microservice @Shipt

Narrated by Vadim Uchitel



About Narrator (https://www.linkedin.com/in/vadim-uchitel-2037b91/)

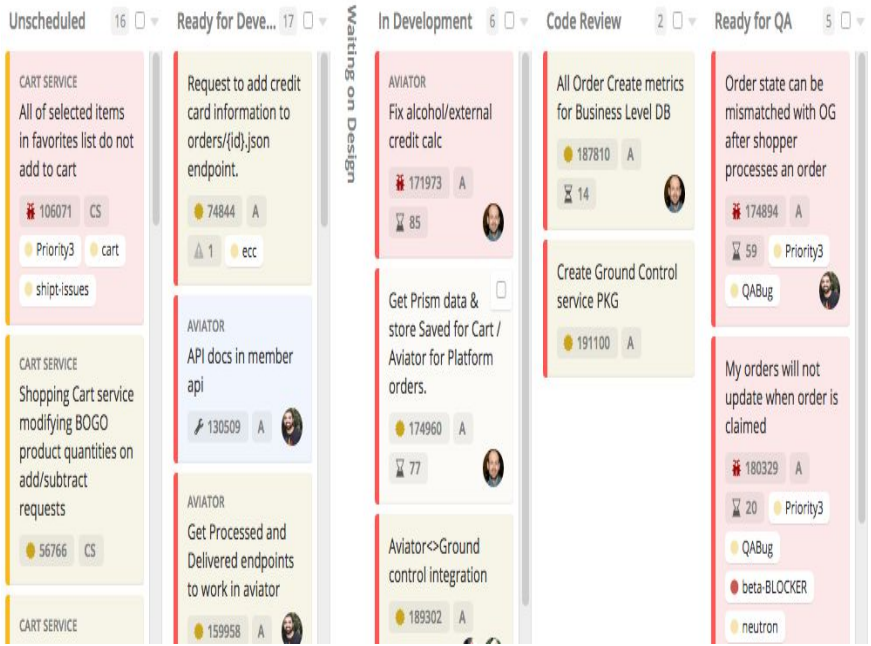
- CFDRC / ESI-CFD (10 years)
  -> Deployed code every 12 months

- SAIC / Leidos (11 years)
  -> Deployed code every 3 - 6 months

- Shipt (2.5 years)
  -> Deploy code several times per day

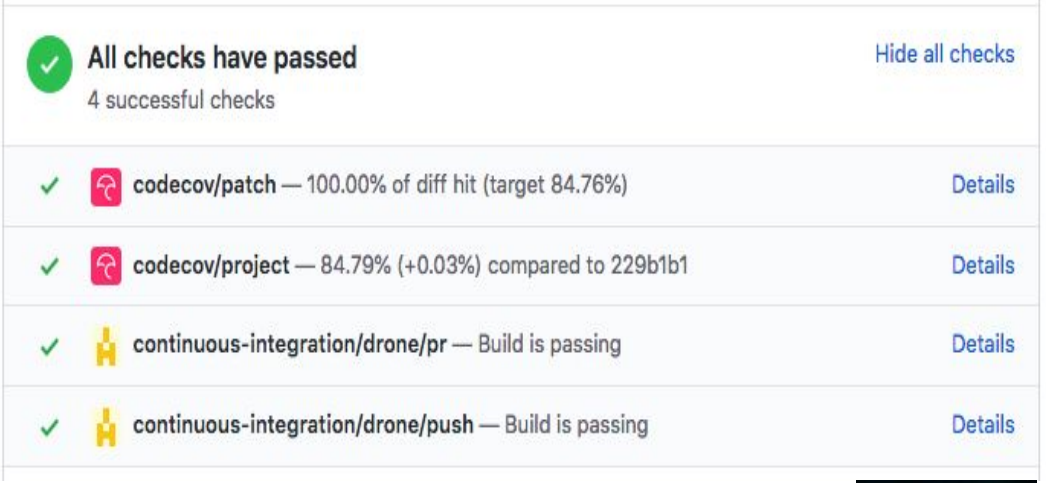
Plan. Clubhouse (JIRA / Kanban ...)

- Creating a card in Clubhouse starts the development process
- Automatic integration with Github
- All status changes are posted to Email / Slack

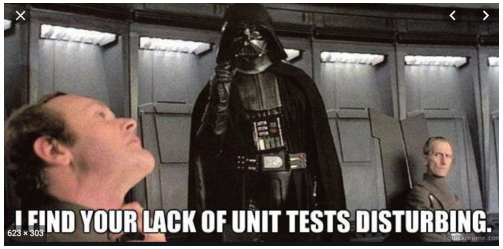


Let’s write code and commit to Github

- Commit code
- Open Pull Request (PR)
- Require 1 or 2 reviews
- Require unit tests (code coverage should go up)
- Require CI to pass
- At Shipt we use Golang (Go), programming language developed by Google



Unit Tests



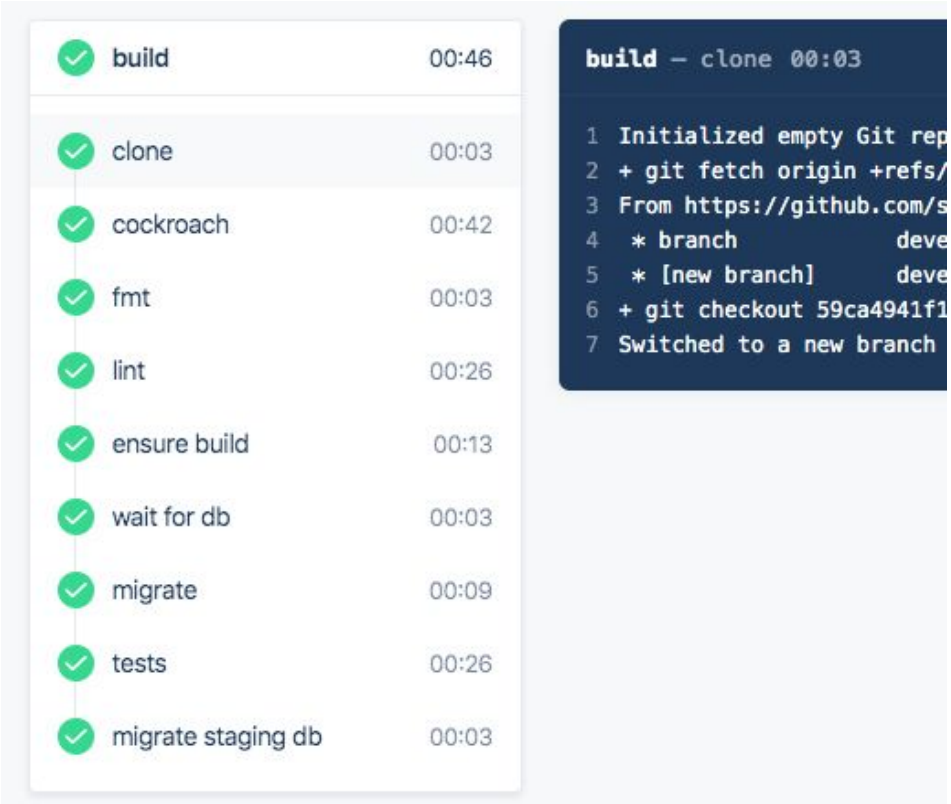
- You can write code faster without unit tests
- You will pay for it in the long term, since making changes to code without unit tests will lead to bugs

-> Pay now or pay more later? Your call

-> At Shipt we require ~85% coverage and use http://codecov.io/ to track it

Continuous Integration (CI). Drone (drone.io)

- Drone vs Jenkins https://rancher.com/drone-vs-jenkins
- “Drone is a wonderful piece of CI software. It has quickly become a very popular choice for wanting to get up and running quickly, looking for a simple container-native CI solution.”
- YAML driven and designed for quick setup of unit tests & building docker containers
- Less powerful than Jenkins with its multiple plugins, but easier to use



Don’t repeat Phoenix. Build Staging environment

- “Staging environments are made to test codes, builds, and updates to ensure quality under a production-like environment before application deployment.”
- Always have a way to mimic production environment. Otherwise, **you will fail**
- At Shipt, Staging infrastructure (but not data) matches Production

Integration Tests

- Integration tests automatically run after each code deploy to Staging
- Integration tests (executed by Drone) call multiple services in Staging environment, validate responses, and notify about failures (via Slack or Email)
- Since services are developed by multiple engineers / teams, integration tests ensure that overall system still operates correctly after the changes

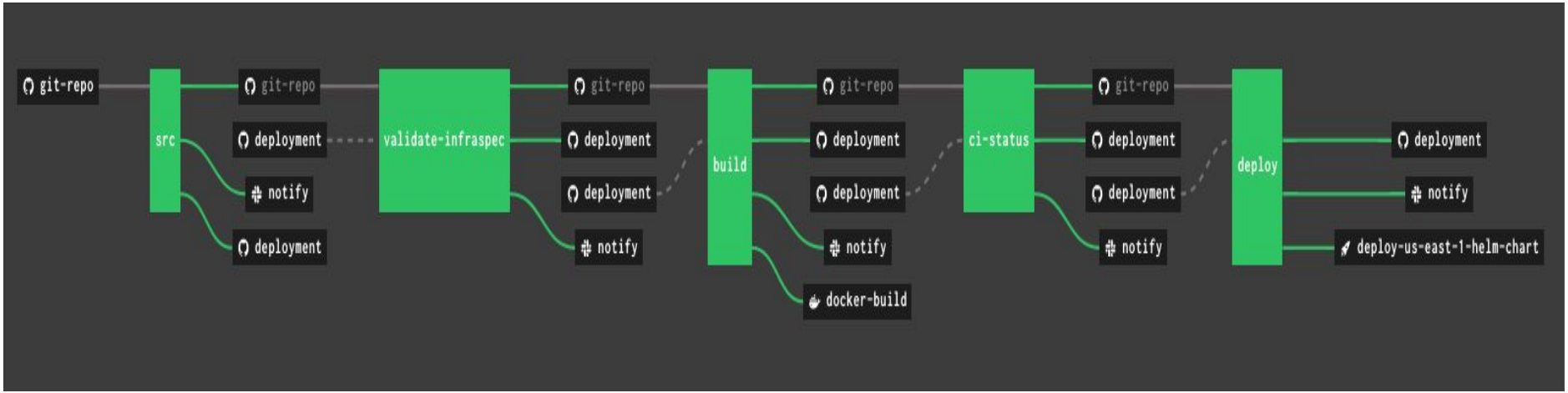
Load Tests

- **Load testing** is performed to determine a system's behavior under both normal and anticipated peak **load** conditions
- At Shipt we use Locust https://github.com/locustio/locust to perform load testing with 10x average production load
- Running load tests allow engineers to sleep better on weekends when system load is at its peak

Continuous Deployment (CD)

Continuous Deployment is a software development practice in which every code change goes through the entire pipeline and is put into production, automatically, resulting in many production **deployments** every day.

Continuous Deployment(CD). Concourse https://concourse-ci.org/

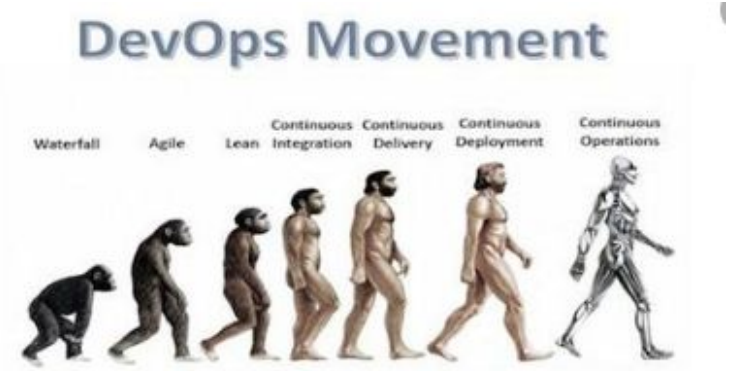


- Concourse provides easy-to-use process and visualization of the deployment pipeline and allows to rerun different parts of the pipeline in the case of the failure
- All pipeline failures & successes are immediately reported to Slack / Email
- Unit tests (run via Drone) are automatically executed and failure will stop the deploy
- Pipeline run (typically takes several minutes) results in code being operational and in-use by Customers
- Pipeline automatically starts after Github code merge to Master branch

Infrastructure as Code

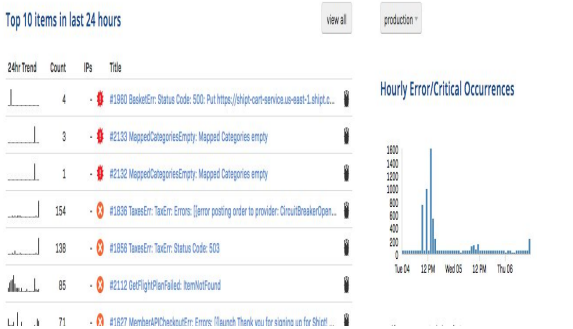
- **All Shipt infrastructure runs on AWS**
- “Infrastructure As Code” allows to provision (as well as delete) necessary cloud resources to run the microservice
- Internal spec (aka Infraspec) https://shipt.tech/infraspec-declarative-microservices-557639222c7c
  - converts YAML file into Terraform script
- Terraform (terraform.io)
  - Allows to specify what your infrastructure should be via scripts
  - Automatically provisions and deletes necessary cloud infrastructure.
  - Works for AWS, GCP, Azure
- New alternative - Pulumi (pulumi.com)
  - Similar to Terraform, but allows to specify your cloud resources in multiple programming languages (Go, JS, Python)

Service is in production
Are we done?



Error Monitoring and Crash reporting

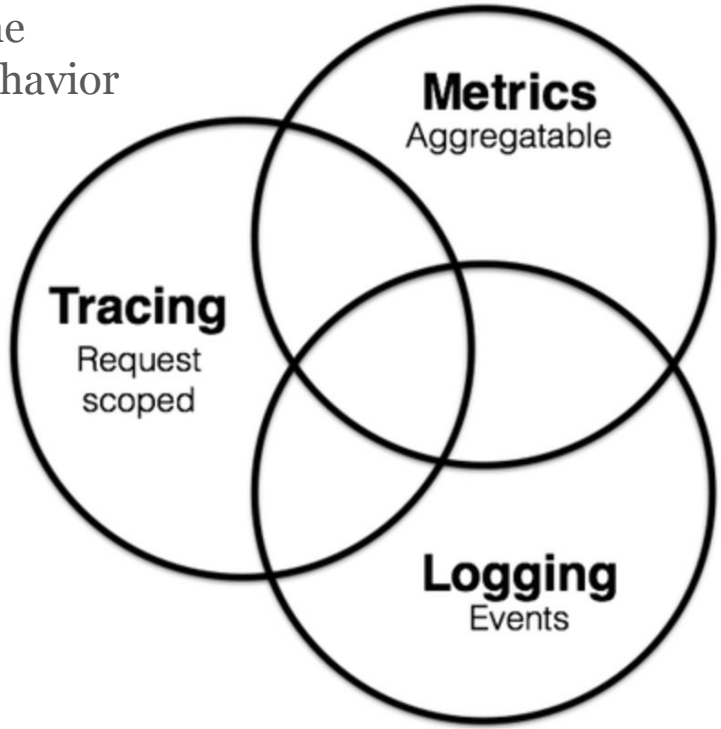
- All critical errors and crashes are reported to Rollbar (rollbar.com) and notifications being sent to Slack / Email
- Rollbar integration with Github allows fast triage and identification of the issue
- All important queues have alerts attached with OpsGenie and Slack alerts



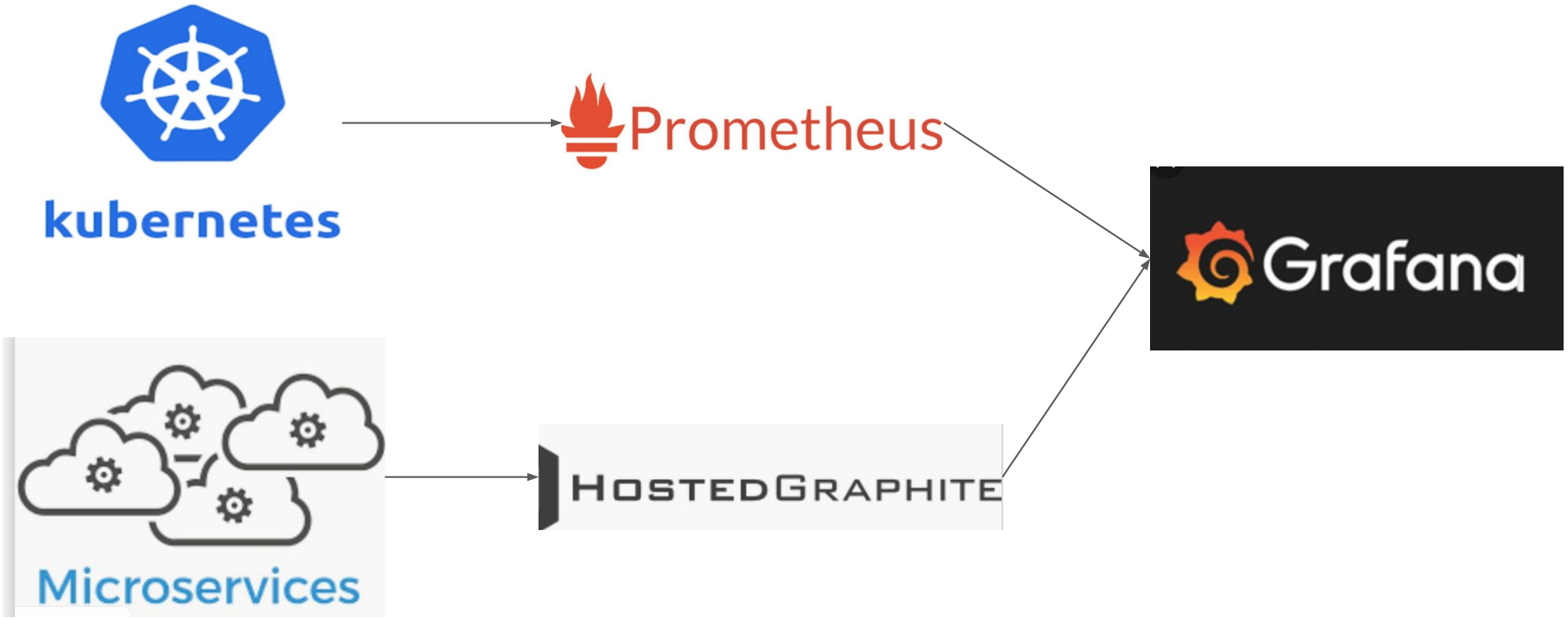

```
Traceback (most recent call last):
  File "vendor/github.com/rollbar/rollbar-go/rollbar.go", line 444 in rollbar-go.Log
  File "vendor/github.com/shipt/shipt-tofu/logger/rollbar.go", line 147 in logger.sendRollbar
  File "vendor/github.com/shipt/shipt-tofu/logger/logger.go", line 87 in logger.trackError
  File "vendor/github.com/shipt/shipt-tofu/logger/logger.go", line 138 in logger.(*InfoLog).Error
  File "vendor/github.com/shipt/shipt-tofu/logger/logger.go", line 260 in logger.Error
  File "internal/controller/helpers.go", line 287 in controller.getFlightPlan
  File "internal/controller/adminController.go", line 77 in controller.AdminController.Get
  File "/usr/local/go/src/net/http/server.go", line 1995 in http.HandlerFunc.ServeHTTP
  File "vendor/github.com/shipt/shipt-tofu/http/server/middleware/timing.go", line 98 in middleware.AliasPath.func1
  File "/usr/local/go/src/net/http/server.go", line 1995 in http.HandlerFunc.ServeHTTP
  File "vendor/github.com/gorilla/mux/mux.go", line 212 in mux.(*Router).ServeHTTP
  File "vendor/github.com/gorilla/handlers/cors.go", line 54 in handlers.(*cors).ServeHTTP
  File "vendor/github.com/urfave/negroni/negroni.go", line 46 in negroni.Wrap.func1
  File "vendor/github.com/urfave/negroni/negroni.go", line 29 in negroni.HandlerFunc.ServeHTTP
  File "vendor/github.com/urfave/negroni/negroni.go", line 38 in negroni.middleware.ServeHTTP
  File "vendor/github.com/shipt/shipt-tofu/http/server/middleware/tracing.go", line 39 in middleware.(*TracingMiddleware).ServeHTTP
  File "vendor/github.com/urfave/negroni/negroni.go", line 38 in negroni.middleware.ServeHTTP
  File "vendor/github.com/urfave/negroni/recovery.go", line 193 in negroni.(*Recovery).ServeHTTP
  File "vendor/github.com/urfave/negroni/negroni.go", line 38 in negroni.middleware.ServeHTTP
  File "vendor/github.com/shipt/shipt-tofu/http/server/middleware/timing.go", line 66 in middleware.(*TimingMiddleware).ServeHTTP
  File "vendor/github.com/urfave/negroni/negroni.go", line 38 in negroni.middleware.ServeHTTP
  File "vendor/github.com/urfave/negroni/negroni.go", line 96 in negroni.(*Negroni).ServeHTTP
  File "/usr/local/go/src/net/http/server.go", line 2774 in http.serverHandler.ServeHTTP
  File "/usr/local/go/src/net/http/server.go", line 1878 in http.(*conn).serve
  File "/usr/local/go/src/runtime/asm_amd64.s", line 1337 in runtime.goexit
```

Observability

Observability is what is required in order to gain understanding of the application and infrastructure behavior in a production environment

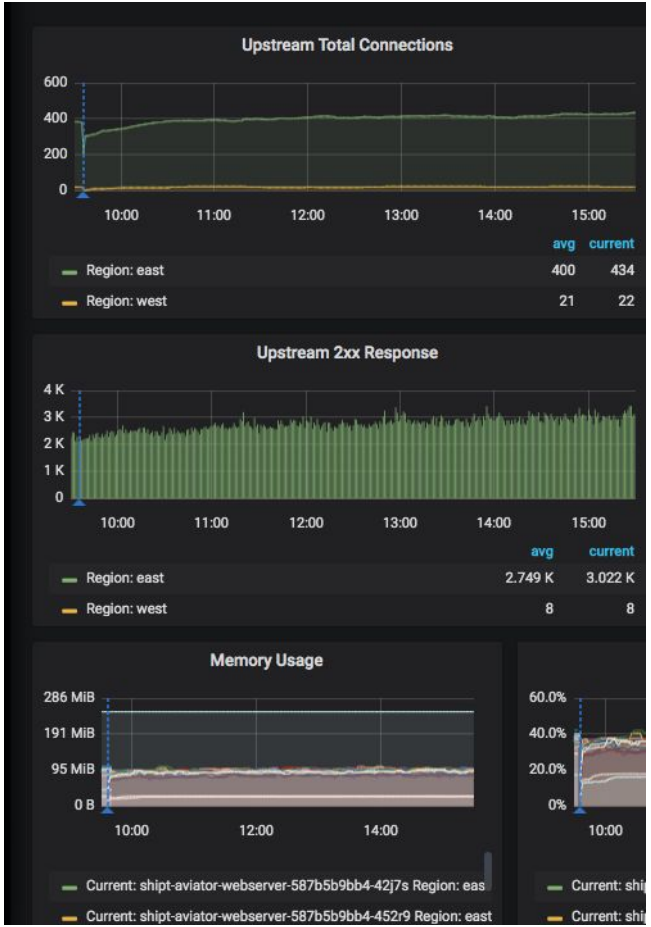


Metrics types (container & service levels)



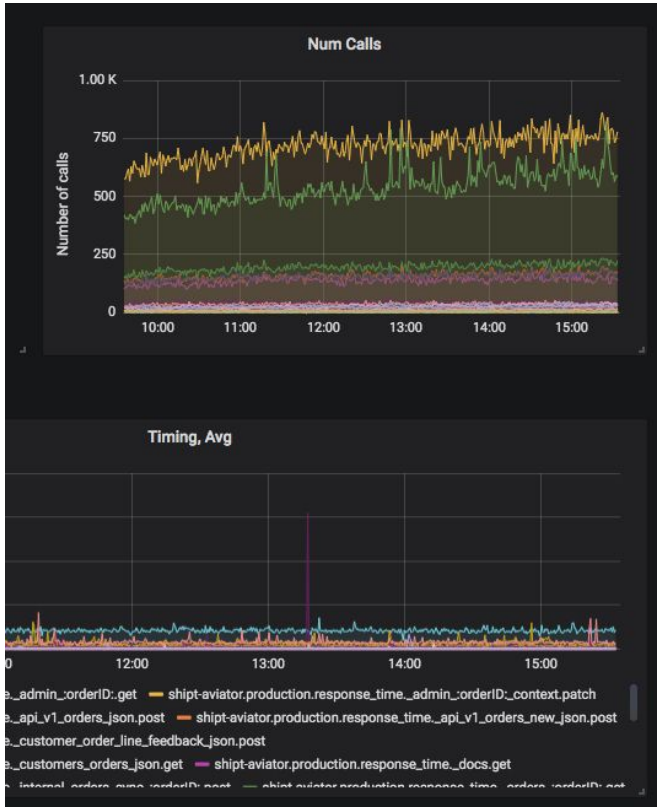
Container-level metrics

- Container-level telemetry provides information about container health wrt to Hardware and Network
- Data provided by:
  - AWS Load Balancers
  - Docker containers
  - K8s tools



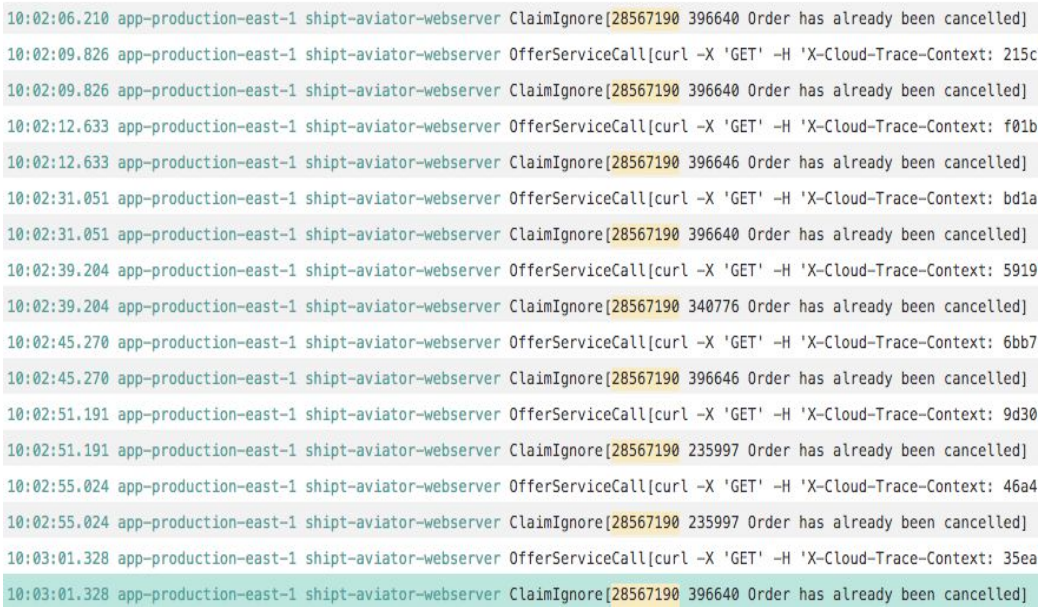
Service-level metrics

- Service level telemetry provides business-specific information that is relevant to a service.
- Examples:
  - Latency data
  - Specific business functions, such as number of XYZ created
  - Number of calls and latency data to access datastores



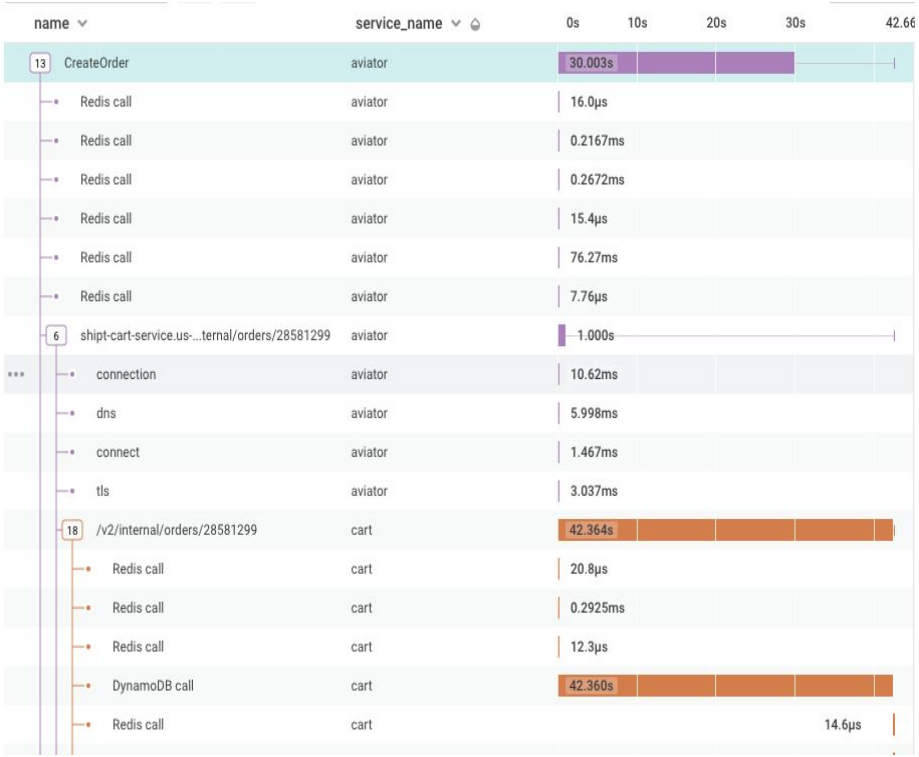
Logging (Scalyr / Splunk)

- Logs (typically taken from container stdout streams or output files) are combined into a single tool for easy searching
- Typically the most useful tool in case of an outage
- Always write your code with ability and control to have as much logging as possible



Tracing / Distributed Tracing (Honeycomb)

- A trace represents a single user's journey through an entire stack
- Allows to track inefficiencies in the system and understand interaction between services
- Tracing tools provide UI to see how request goes through the system

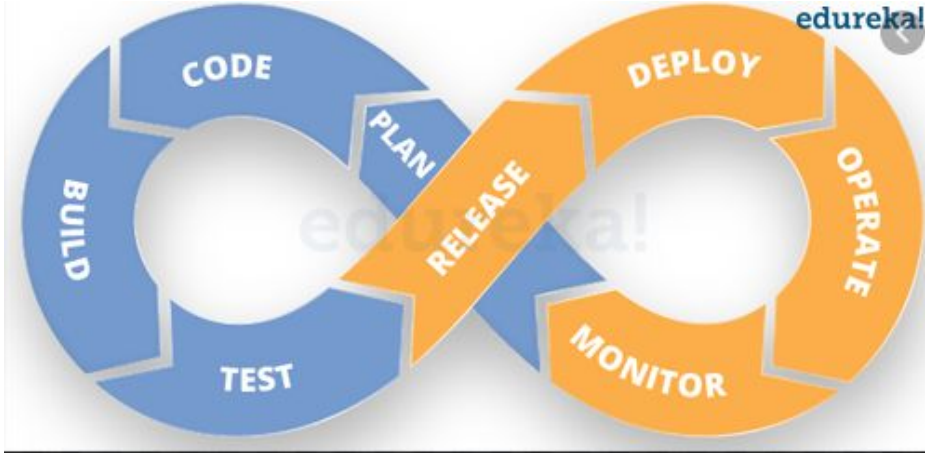


Close the loop and learn from failures

- No blame Post-Mortems / Retrospectives
  - Each incident is followed by no blame post-mortem and creation of action items and new Clubhouse cards
- Always dedicate time to repay tech debt


DEV
OPS

Questions?

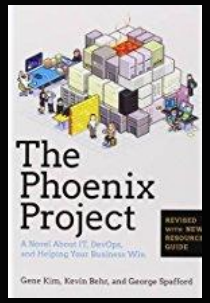

edureka!
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR

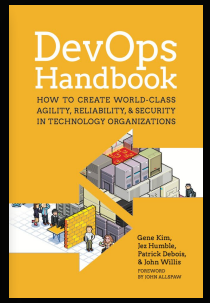
DevOps Resources

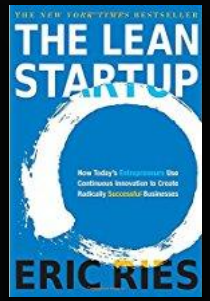
https://devopsfordefense.org/resources/

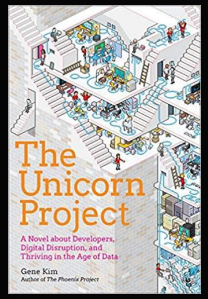
https://www.meetup.com/DevOps-for-Defense/
https://github.com/jondavid-black/DevOpsForDefense
devopsfordefense@gmail.com

Books / Publications:





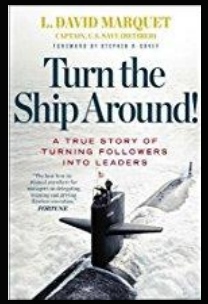


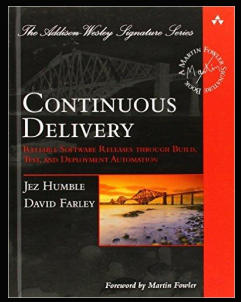


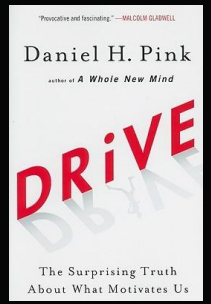
Conference Presentations (YouTube):

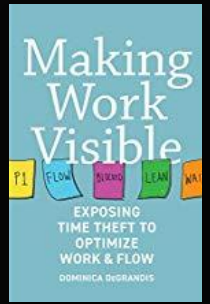
- DevOps Enterprise Summit (DOES)
- IT Revolution
- Velocity
- GoTo

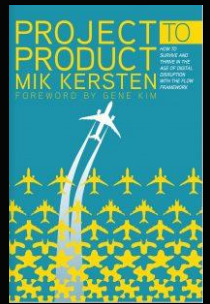
YouTube

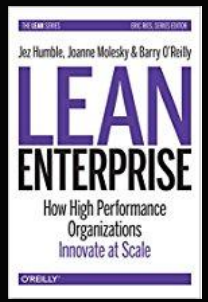


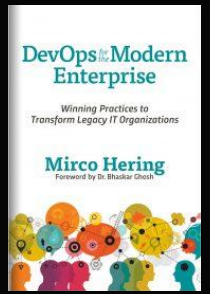


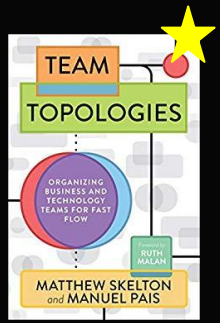












Group Exercise: Lean Coffee

Book Club: The Unicorn Project

1. Each table has a facilitator.
2. The facilitator has a short introduction.
3. Everyone write down questions or topics for discussion on the subject. Place them in the middle of the table.
4. The group votes on each question or topic by placing a dot on the card. 3 votes per person.
5. Cards with most dots goes first. Set a timer for 5 minutes and discuss.
6. After 5 minutes, either vote (thumbs up/down) to keep going or move on to the next card.

Suggested Topics: “Team Interactions”, “Better Tools”, “#NoMeetingFriday”