CHAOS ENGINEERING BOOTCAMP

Ana Medina
Chaos Engineer at Gremlin

SRECon EMEA - August 2018

Ana Medina

Ana is currently working as a **Chaos Engineer** at Gremlin, helping companies avoid outages by running proactive chaos engineering experiments. She last worked at Uber where she was an engineer on the SRE and Infrastructure teams specifically focusing on chaos engineering and cloud computing.

@ana_m_medina

CHAOS ENGINEERING BOOTCAMP

Gremlin

Chaos Engineer /ˈkāˌäs /ˌenjəˈnir/
noun

1. a person helping companies avoid outages by running proactive chaos engineering experiments.

gremlin.com
@gremlininc

CHAOS ENGINEERING BOOTCAMP

Ho Ming Li

Ho Ming Li is the **Lead Solutions Architect** at Gremlin. Prior to joining Gremlin, he worked at Amazon Web Services with many customers providing guidance around architectural and operational best practices. He takes a strategic approach to deliver holistic solutions, often diving into the intersection of people, process, business, and technology. His goal is to enable everyone to build more resilient software by means of Chaos Engineering practices.

@HoReaL

CHAOS ENGINEERING BOOTCAMP

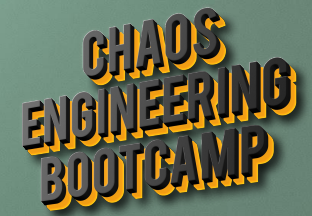
Join Slack

www.gremlin.com/slack
#srecon18_europe

CHAOS ENGINEERING BOOTCAMP

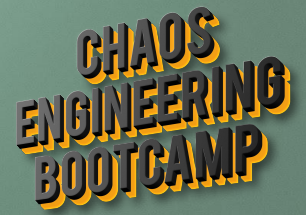
Agenda:

14:00 - 15:00		Foundation of Chaos Engineering
		Breaking Things
15:00 - 15:30		Chaos Engineering Discussion
15:30 - 16:00		*Break*
16:00 - 16:30		Distributed Systems Chaos
16:30 - 16:45		Crafting your own experiment
16:45 - 17:00		Starting at your company
17:00 - 17:15		Advanced Chaos
17:15 - 17:30	❓	Q&A



Part 1️⃣:

Foundations of Chaos Engineering



Chaos Engineering?

Thoughtful, planned experiments designed to reveal the weakness in our systems.

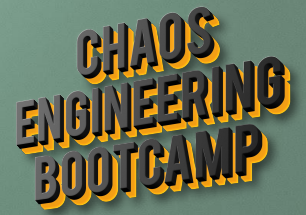
Like a vaccine, we inject harm

to build ***immunity***.

CHAOS ENGINEERING BOOTCAMP

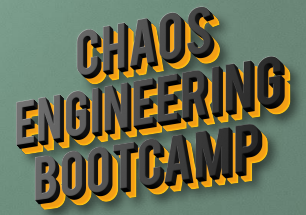
What Chaos Engineering is

- Thoughtful chaos engineering experiments
- Controlled and planned chaos engineering experiments
- Preparing for unpredictable failure
- Preparing engineers for failure
- Preparing for GameDay
- A way to improve SLA
  - fortify systems
  - build and move fast
  - build confidence in systems
  - reveal weak points in your systems
  - build assurance that you can still serve your customers



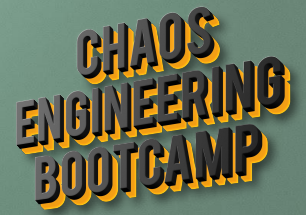
What Chaos Engineering is not

- Random chaos engineering experiments
- Unsupervised chaos engineering experiments
- Unmonitored chaos engineering experiments
- Unexpected chaos engineering experiments
- Breaking production by accident
- Creating outages



Why do Chaos Engineering

- Microservice Architecture is tricky
- Our systems are scaling fast
- Services will fail
- Dependencies on other companies will fail
- Prepares for real world scenarios
- Reduce amount of outages, reduce down time, lose less money

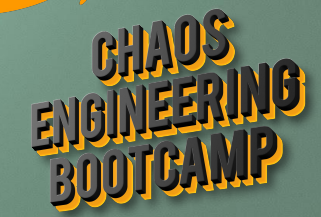


Inject Failure at any level

- Application
- API
- Caching
- Database
- Hardware
- Cloud Infrastructure / Bare metal

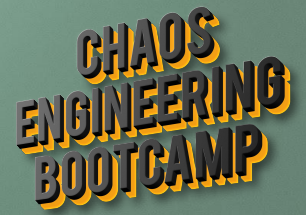
Top places to inject chaos

Companies doing Chaos Engineering



What do you need before doing Chaos Engineering?

- Monitoring / Observability
- On Call / Incident Management
- Alerting and paging
- Clear instructions on how to roll back an experiment
- The cost of downtime per hour



Monitoring and Measuring

- System Metrics: CPU, Disk, I/O
- Availability
- Service specific KPIs
- Customer complaints

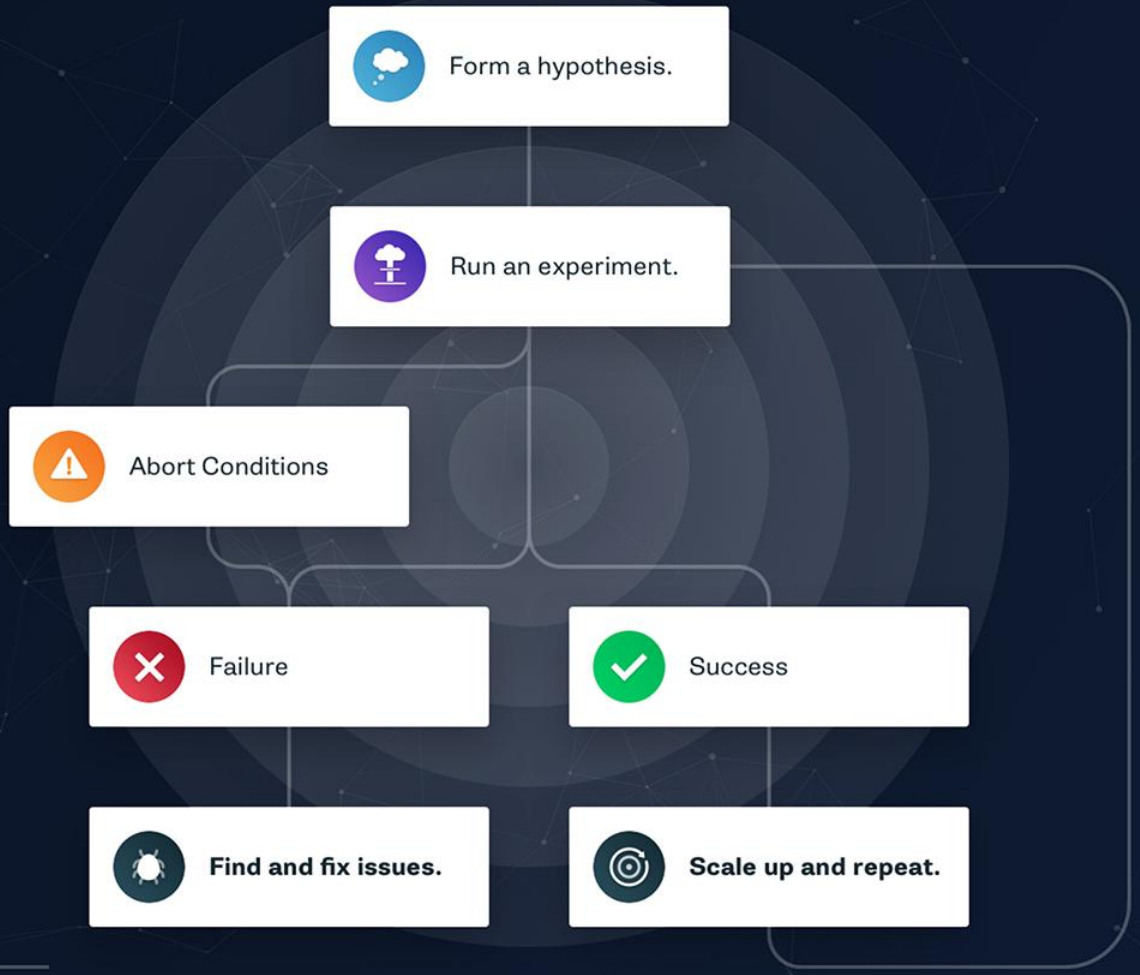
Some Monitoring / Observability tools to use

CHAOS ENGINEERING BOOTCAMP

Part 2️⃣:

Breaking Things

Minimize the blast radius.

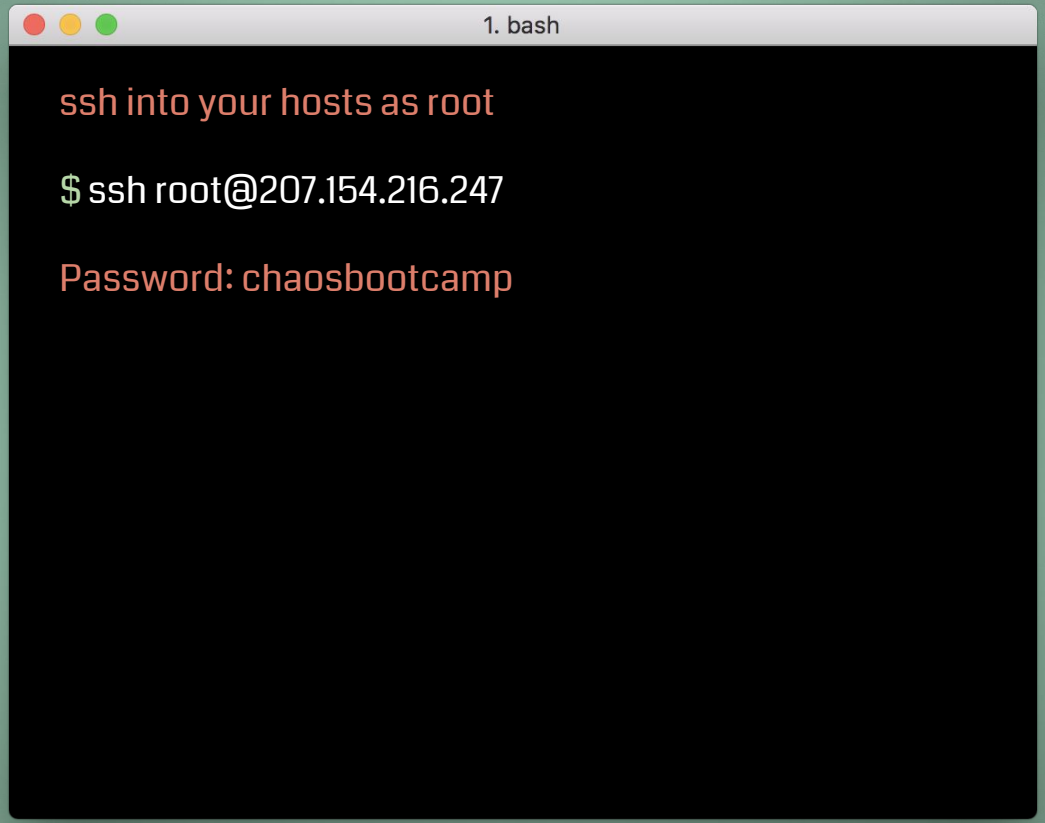


Please get in groups of 3

Getting Access to your host:

While we wait, check out this survey:

bit.ly/ce-questions

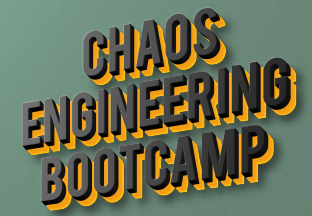


CHAOS ENGINEERING BOOTCAMP

1. bash

Install kubernetes on all hosts

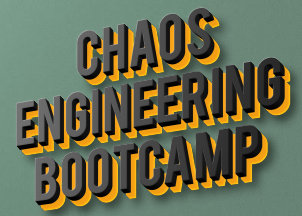
```
apt-get update && apt-get install -y apt-transport-https
curl -s
https://packages.cloud.google.com/apt/doc/apt-key.gp
g | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
apt-get update
apt-get install -y kubelet kubeadm kubectl docker.io
```

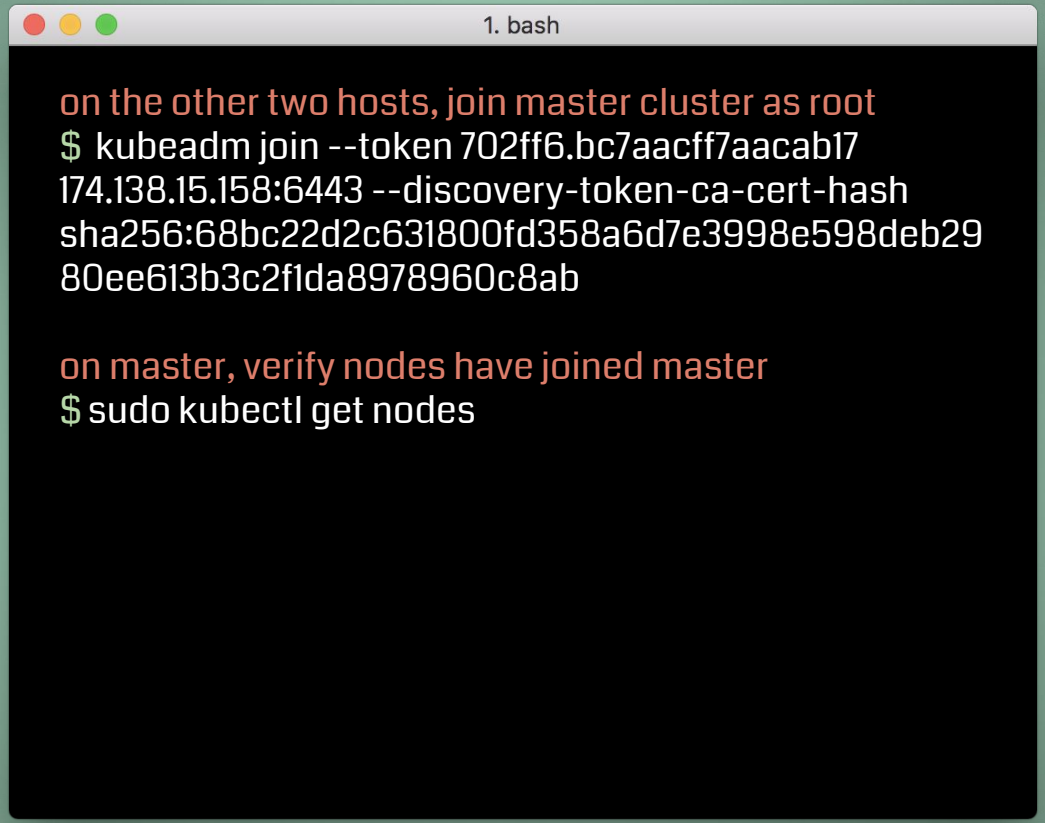


1. bash

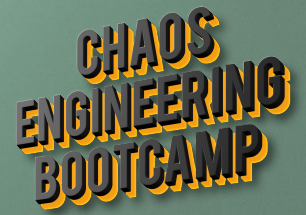
choose host that will be master

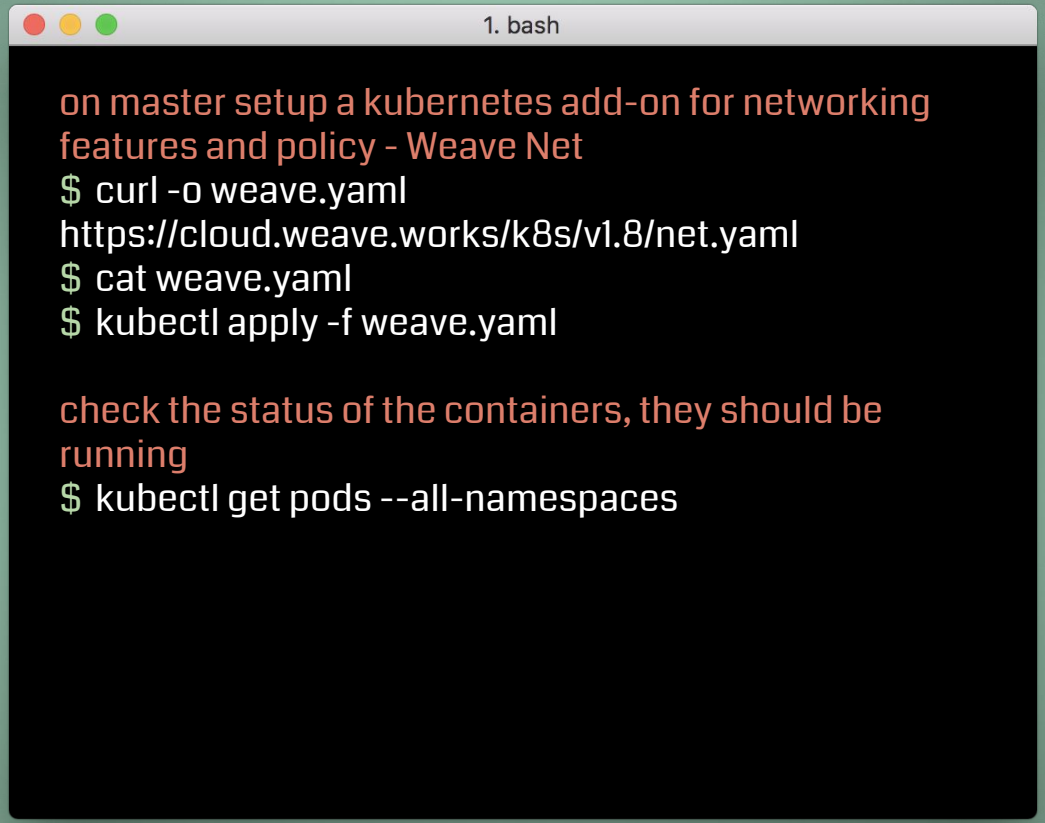
```
$ su -l kube
Password: chaosbootcamp

initialize kubernetes master node
$ sudo kubeadm init

start cluster on master
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf
$HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```




1. bash
on the other two hosts, join master cluster as root
$ kubeadm join --token 702ff6.bc7aacff7aacab17
174.138.15.158:6443 --discovery-token-ca-cert-hash
sha256:68bc22d2c631800fd358a6d7e3998e598deb29
80ee613b3c2f1da8978960c8ab
on master, verify nodes have joined master
$ sudo kubectl get nodes

cool! you've gotten Kubernetes set up, lets setup the rest

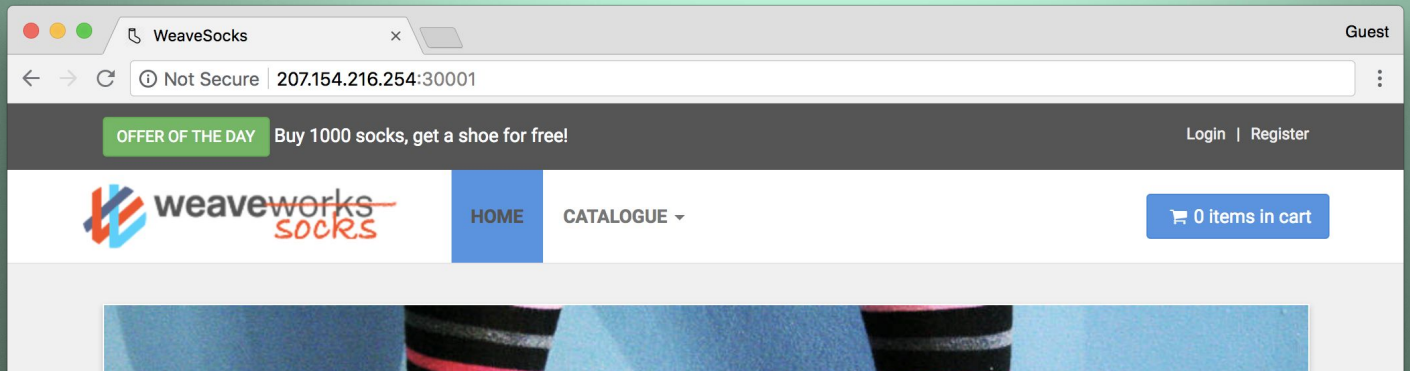



1. bash
on master setup a kubernetes add-on for networking features and policy - Weave Net
$ curl -o weave.yaml https://cloud.weave.works/k8s/v1.8/net.yaml
$ cat weave.yaml
$ kubectl apply -f weave.yaml
check the status of the containers, they should be running
$ kubectl get pods --all-namespaces

1. bash

```
still on master, grab demo microservices sock shop
$ git clone
https://github.com/microservices-demo/microservices-
demo.git
$ cd microservices-demo/deploy/kubernetes/

create namespace
$ kubectl create namespace sock-shop

appy demo to the cluster
$ kubectl apply -f complete-demo.yaml

check if pods are running
$ kubectl get pods --namespace sock-shop
```

hooray! things are running. visit your ip address on port 30001



CHAOS ENGINEERING BOOTCAMP

1. bash

On every host as root, lets get some monitoring in place with Datadog

```
$ DD_API_KEY=faff9c88d8cdd357d76505f595f23797 bash -c "$(curl -L https://raw.githubusercontent.com/DataDog/datadog-agent/master/cmd/agent/install_script.sh)"
```

on every host, let's grab the bootcamp files

```
$ su - chaos
$ git clone https://github.com/tammybutow/chaosengineeringbootcamp
```

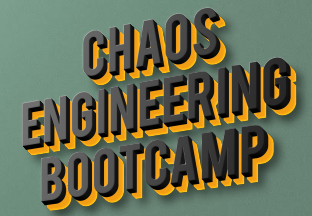
CHAOS ENGINEERING BOOTCAMP

let the chaos begin
hello world, the chaos way

1. bash

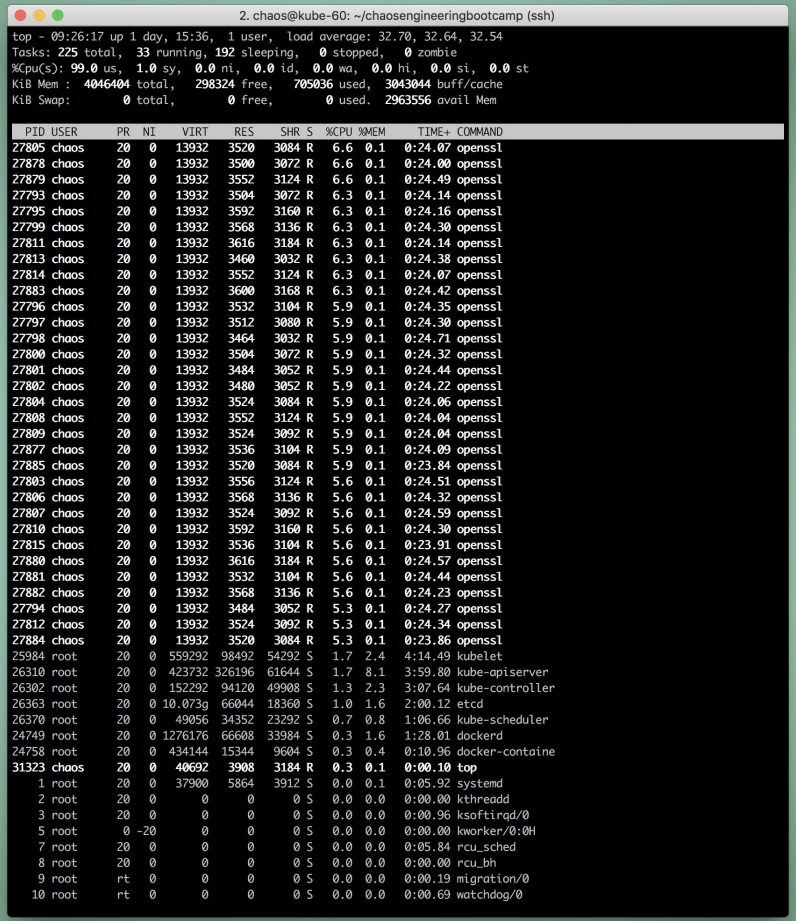
as chaos user on every host
$ cd chaosengineeringbootcamp
$ ls
$ chmod +x chaos_cpu.sh
$ ./chaos_cpu.sh

you can exit using control + c
let's see what we did
$ top

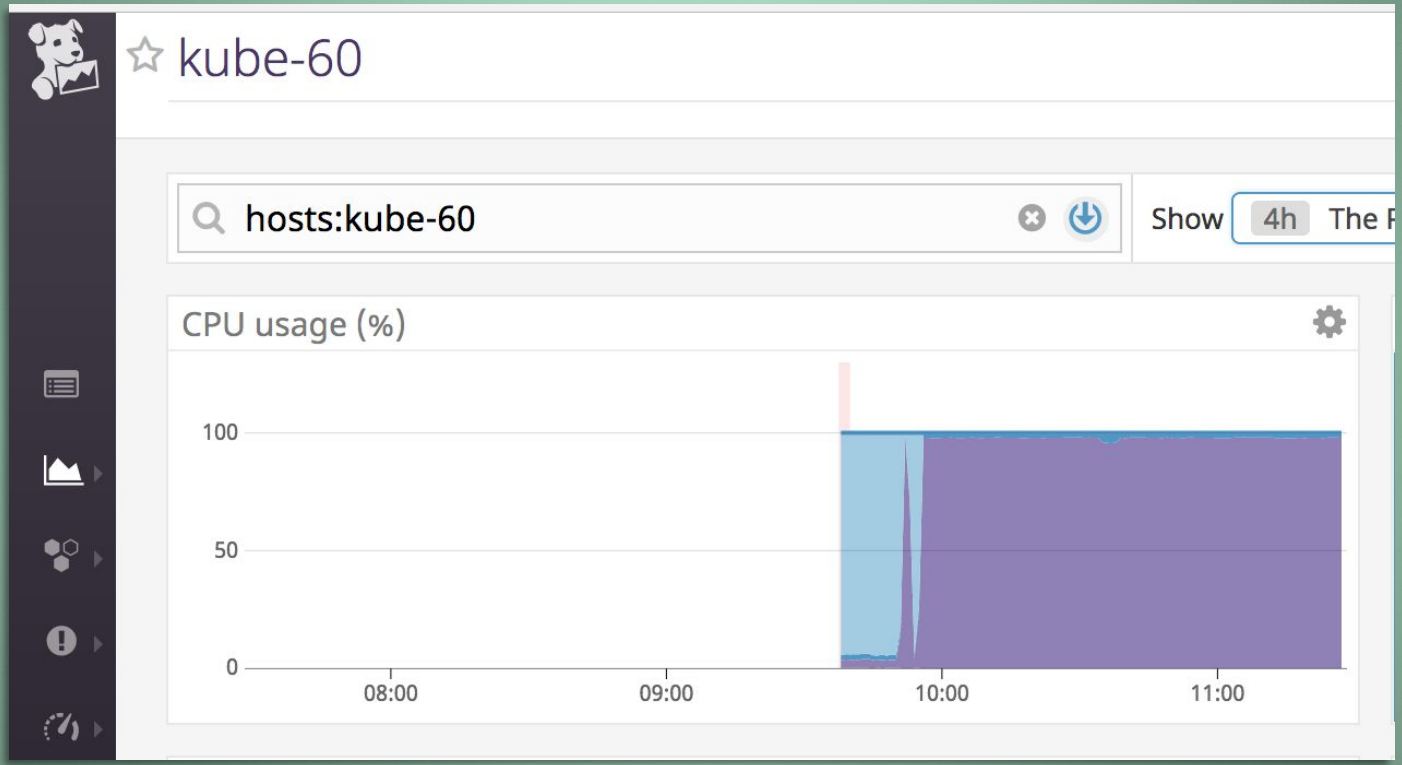


#SRECon

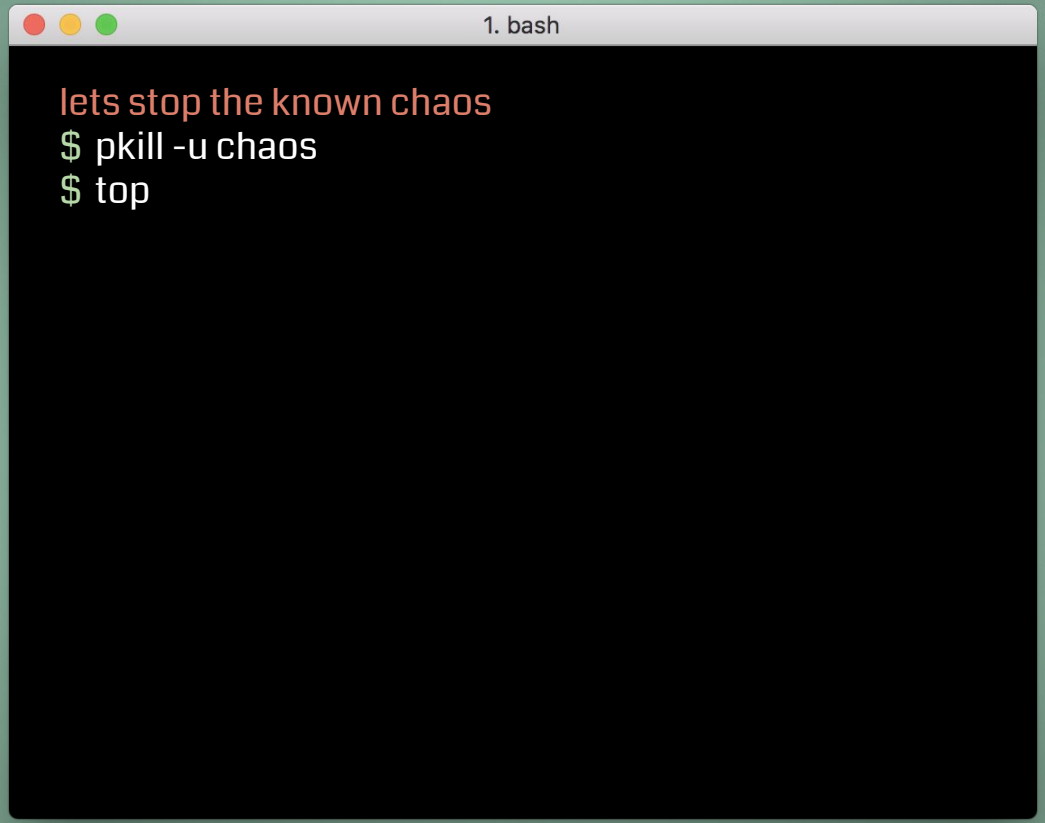
@ana_m_medina



CHAOS ENGINEERING BOOTCAMP

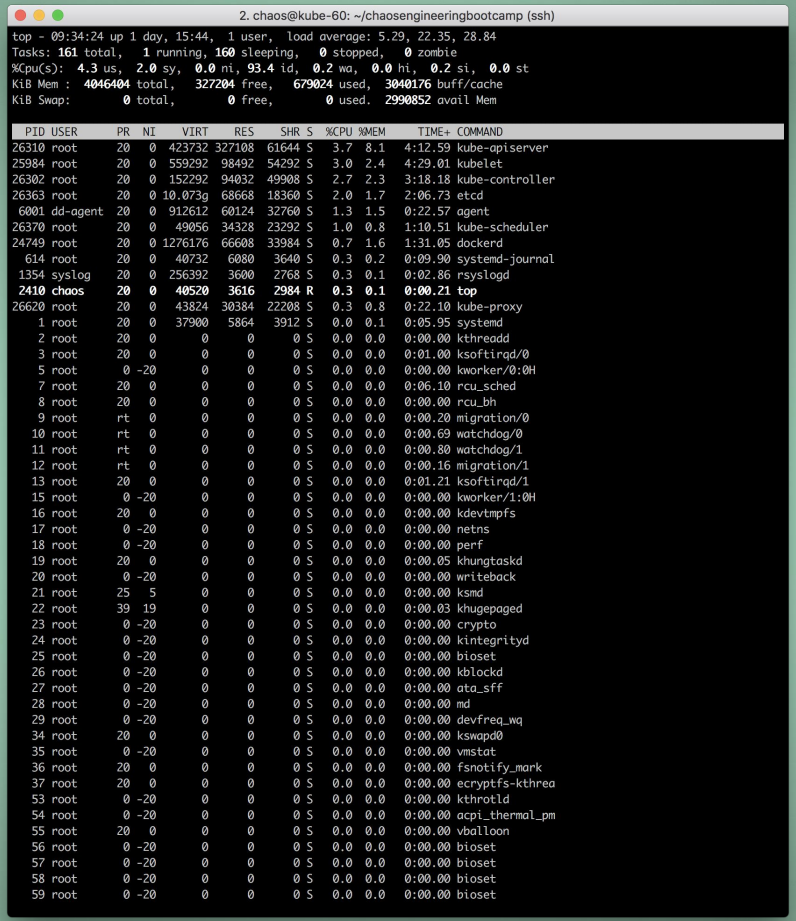


CHAOS ENGINEERING BOOTCAMP


1. bash
lets stop the known chaos
$ pkill -u chaos
$ top

#SRECon

@ana_m_medina



CHAOS ENGINEERING BOOTCAMP

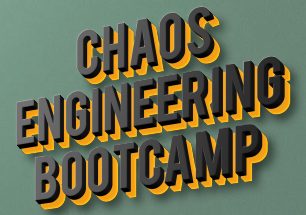
Part 3️⃣:

Chaos Engineering Discussion

4 volunteers needed for 2 teams

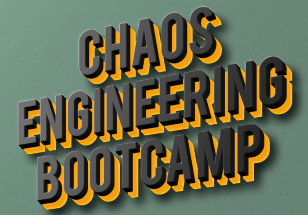
Every company should be doing chaos engineering.

- 2 minutes for brainstorming
- 2 minutes per speaker
- Winning team chosen via applauses



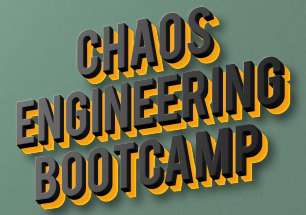
Agenda:

14:00 - 15:00	 Foundation of Chaos Engineering
	 Breaking Things
15:00 - 15:30	 Chaos Engineering Discussion
15:30 - 16:00	 ***Break***
16:00 - 16:30	 Distributed Systems Chaos
16:30 - 16:45	 Crafting your own experiment
16:45 - 17:00	 Starting at your company
17:00 - 17:15	 Advanced Chaos
17:15 - 17:30	 Q&A



Part 4️⃣:

Distributed Systems Chaos



Chaos Monkey

Chaos Monkey is a resiliency tool that helps applications tolerate random instance failures.

github.com/Netflix/chaosmonkey

Simian Army

Tools for keeping your cloud operating in top form. Chaos Monkey is a resiliency tool that helps applications tolerate random instance failures.

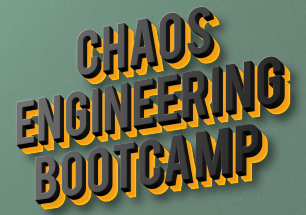
github.com/Netflix/SimianArmy

CHAOS ENGINEERING BOOTCAMP

Kube Monkey

An implementation of Netflix's Chaos Monkey for Kubernetes clusters

github.com/asobti/kube-monkey



Pumba

Chaos testing and network emulation tool for Docker.

github.com/alexei-led/pumba

Powerfulseal

A powerful testing tool for Kubernetes clusters.

github.com/bloomberg/powerfulseal

Litmus

Litmus is chaos engineering for stateful workloads on Kubernetes, hopefully without learning curves

github.com/openebs/litmus

Gremlin

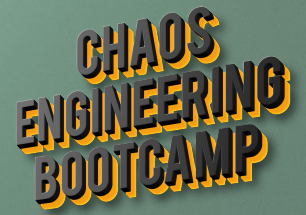
Failure as a Service.
Finds weaknesses in your system before they cause problems.
Run Gremlin Agents on Hosts or Containers. Schedule attacks using UI, API and CLI. Provides 11 attacks out of the box

gremlin.com

CHAOS ENGINEERING BOOTCAMP

Why do Chaos Engineering?

prevent outages



Northeast blackout of 2003 - Wikipedia — Guest

Secure https://en.wikipedia.org/wiki/Northeast_blackout_of_2003

Not logged in Talk Contributions Create account Log in

Article Talk | Read Edit View history | Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Northeast blackout of 2003

From Wikipedia, the free encyclopedia

This article has multiple issues. Please help **improve it** or discuss these issues [hide] on the **talk page**. *(Learn how and when to remove these template messages)*

- This article **has an unclear citation style**. *(May 2013)*
- This article **needs additional citations for verification**. *(July 2008)*

The **Northeast blackout of 2003** was a widespread power outage throughout parts of the Northeastern and Midwestern United States and the Canadian province of Ontario on August 14, 2003, just after 4:10 p.m. EDT.[1]

Some power was restored by 11 p.m. Most did not get their power back until two days later. In other areas, it took nearly a week or two for power to be restored.[2] At the time, it was the world's second most widespread blackout in history, after the 1999 Southern Brazil blackout.[3][4] The outage, which was much more widespread than the Northeast Blackout of 1965, affected an estimated 10 million people in Ontario and 45 million people in eight U.S. states.

The blackout's primary cause was a software bug in the alarm system at the control room of FirstEnergy Corporation, an Akron, Ohio–based company, causing operators to remain unaware of the need to re-distribute load after overloaded transmission lines drooped into foliage. What should have been a manageable local blackout cascaded into collapse of the entire electric grid.

Contents [hide]

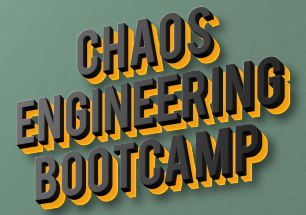
1 Immediate impact
  1.1 Unaffected regions
2 Causes
  2.1 Background

This image shows states and provinces that experienced power outages. Not all areas within these political boundaries were affected.

Northeast blackout of 2003

System operators were unaware of the malfunction. The failure deprived them of both audio and visual alerts for important changes in system state. Race condition triggered in the control software.

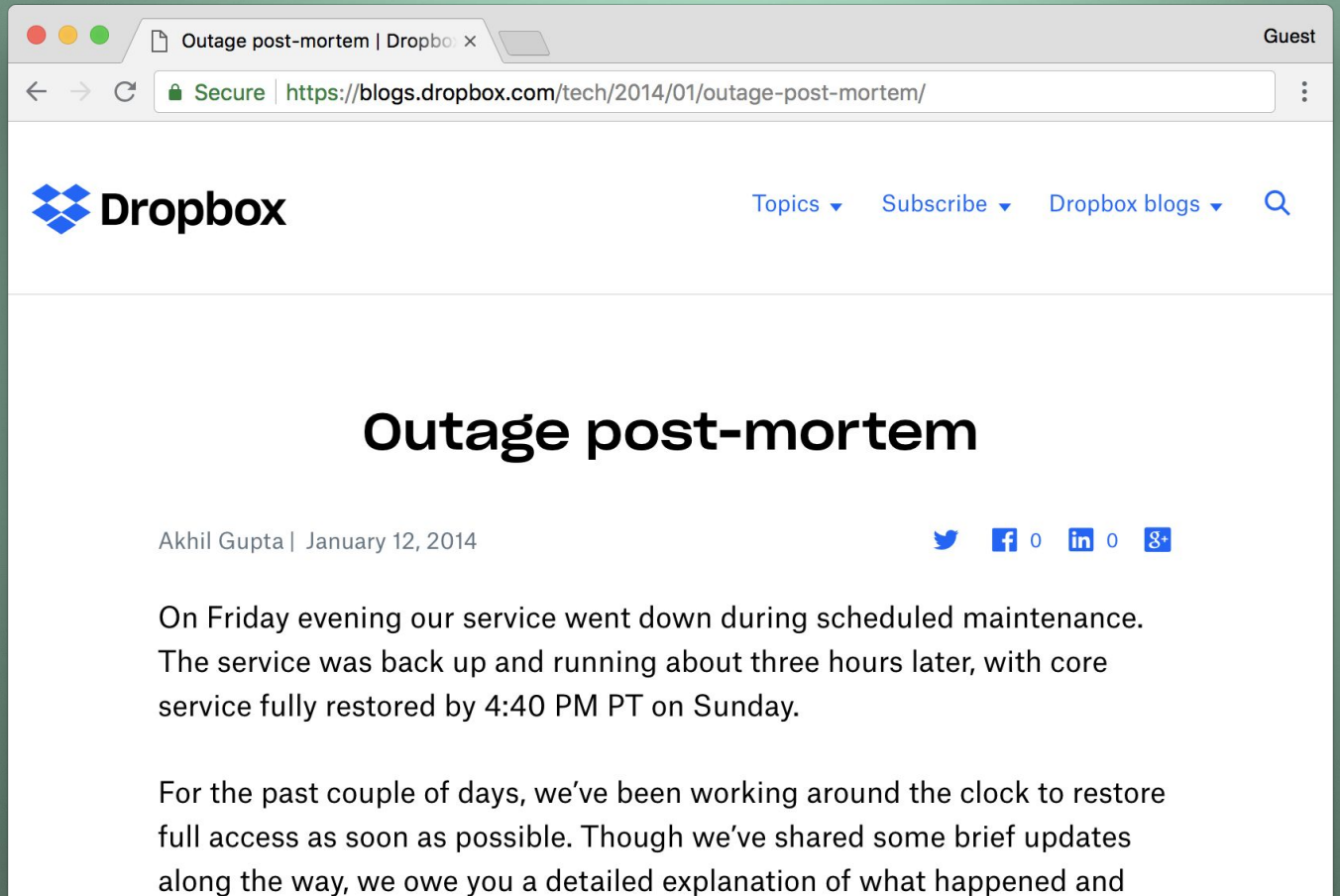
Chas Engineering helps you test monitoring tools, metrics, dashboards, alerts, and thresholds.



Learn about different outages:

github.com/danluu/post-mortems

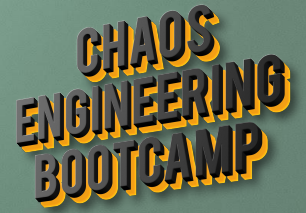
Injecting Chaos is a controlled way will lead to engineers building resilient systems.

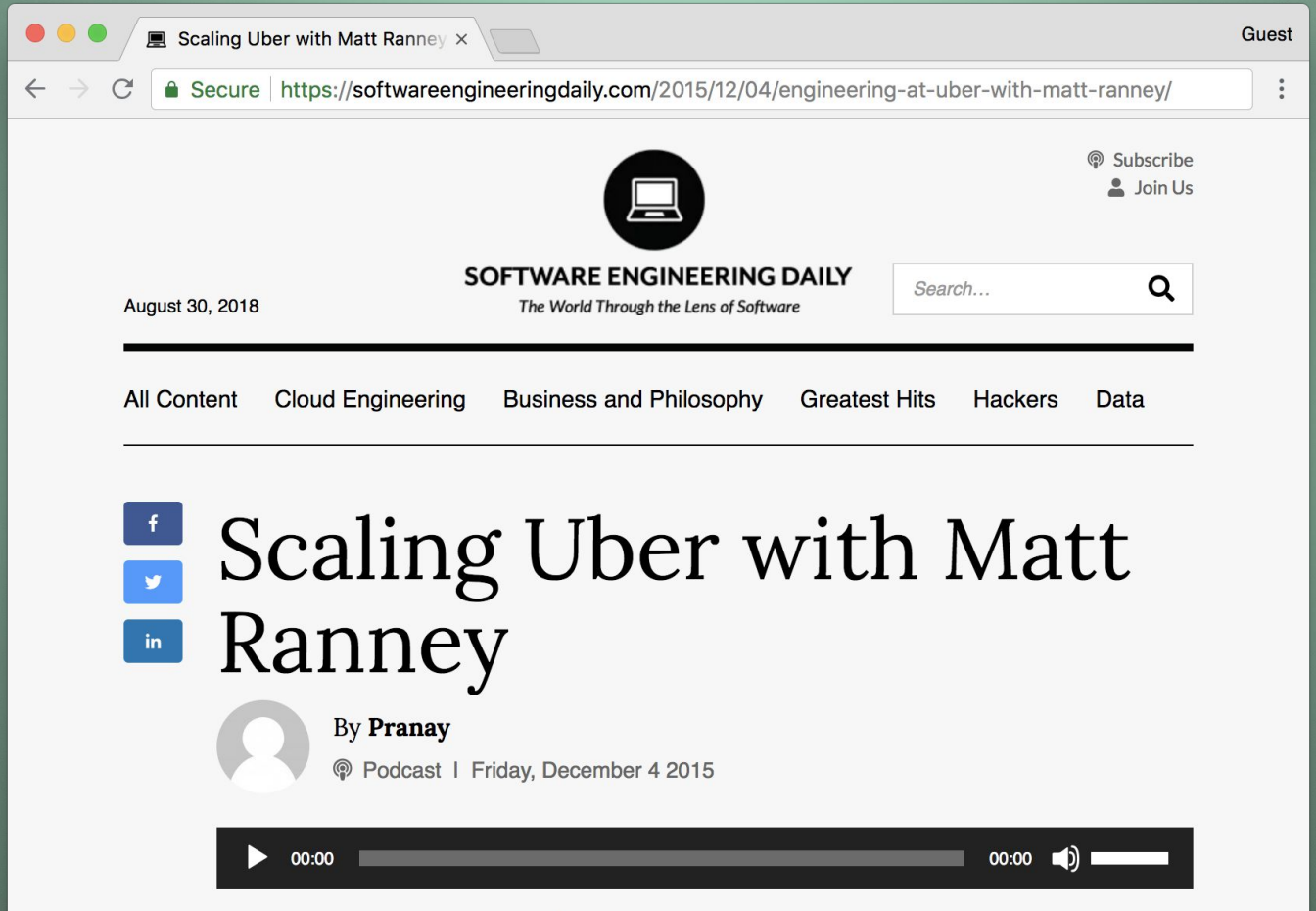


CHAOS ENGINEERING BOOTCAMP

Some master-replica pairs were impacted which resulted in site going down

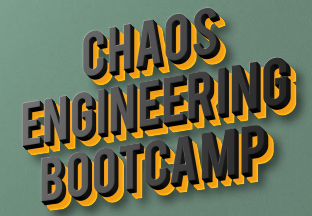
https://blogs.dropbox.com/tech/2014/01/outage-post-mortem/




Scaling Uber with Matt Ranney
Guest
Secure https://softwareengineeringdaily.com/2015/12/04/engineering-at-uber-with-matt-ranney/
Subscribe
Join Us
August 30, 2018
SOFTWARE ENGINEERING DAILY
The World Through the Lens of Software
Search...
All Content
Cloud Engineering
Business and Philosophy
Greatest Hits
Hackers
Data
Scaling Uber with Matt Ranney
By Pranay
Podcast | Friday, December 4 2015
00:00
00:00

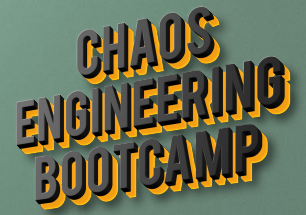
Uber's Database Outage

Master log replication to S3 failed
Logs backed up on primary
Alerts were fired and ignored
Disk fills up on database primary
Engineer deletes unarchived WAL files
Error in config prevents promotion

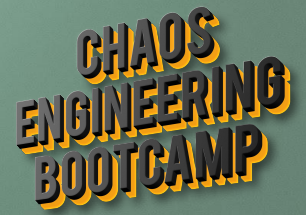


Part 5️⃣:

Crafting your own experiment



Don’t approach it with a random strategy, instead approach it like a scientific experiment, thoughtful and planned.



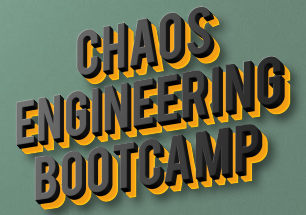
Crafting your own experiment

Brainstorming: What should we break?

1. Form a hypothesis - What could go wrong?
2. Plan your experiment
3. Minimize Blast Radius - Small experiments first
4. Run experiment
5. Observe results

    if things broke -> go fix it.

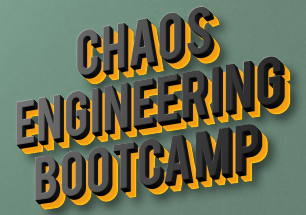
    If things went as planned, increase blast radius and go back to step #1



Part 6️⃣:

Starting at your company

Are you confident about your metrics and alerting?

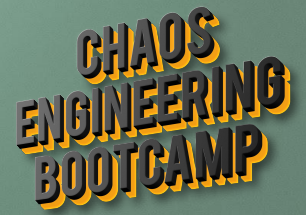


Are you confident your customers are getting as good as an experience as they should be?

Are you losing money due to downtime and broken features?

Getting Started:

1. Identify top 5 critical systems
2. Choose system
3. Whiteboard the system
4. Determine what experiment you want to run: (resource, state, network)
5. Determine Blast Radius

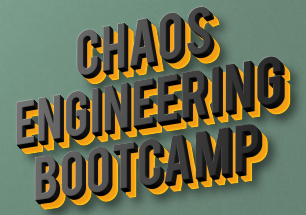


GameDay

Chose a system / application

Allocate 2-4 hours

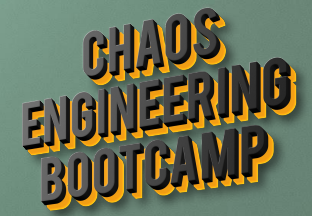
Room filled with the engineers that developed / support application



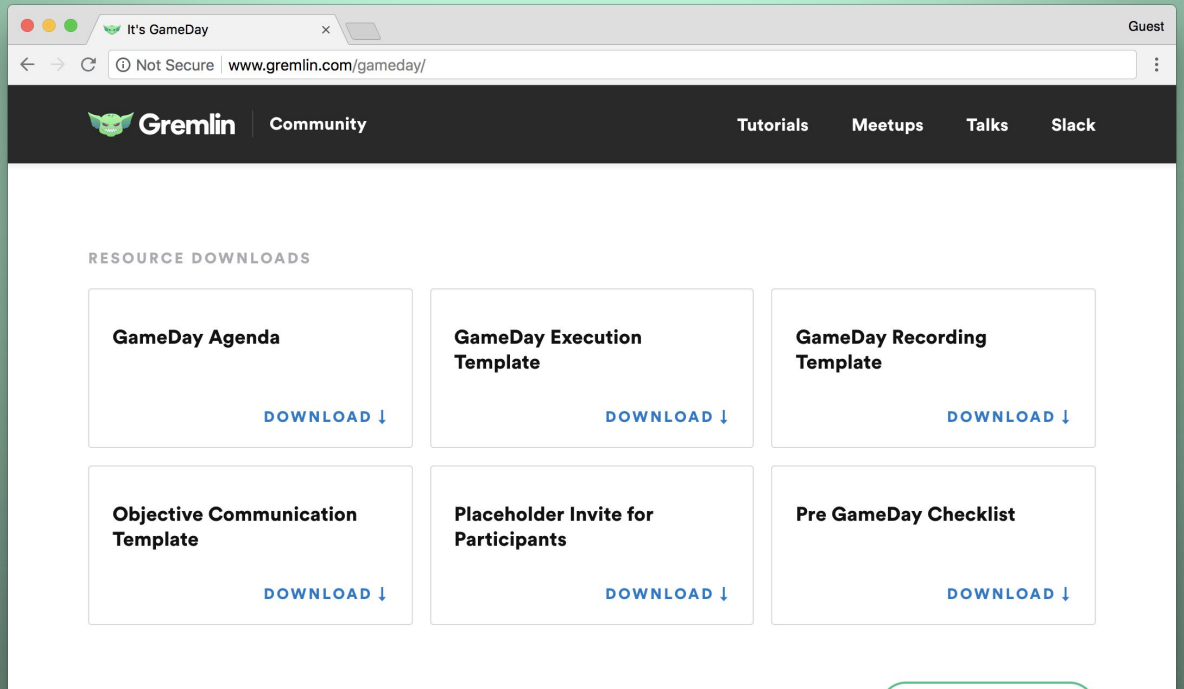
GameDay

"What could go wrong?"

"Do we know what will happen if this breaks?"



gremlin.com/gameday



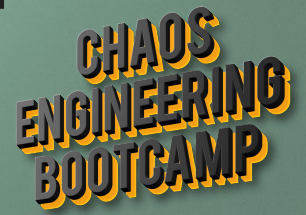
CHAOS ENGINEERING BOOTCAMP

Failure Fridays

Dogfooding.

We use Gremlin for Chaos Engineering Experiments

Dedicated time to practice Chaos Engineering to reveal weaknesses in our services

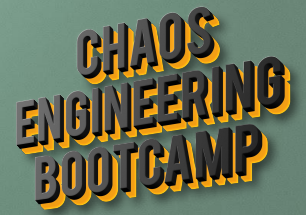


Part 7️⃣:

Advanced Chaos

Other Uses for Chaos Engineering

- Preparing for IPO / Auditing
- Training new engineers / on-call practicing
- Testing alert thresholds



Disorderly Labs

Guest

https://disorderlylabs.github.io

Disorderly Labs Research Projects Publications Talks Team Press Contact Us

DISORDERLY

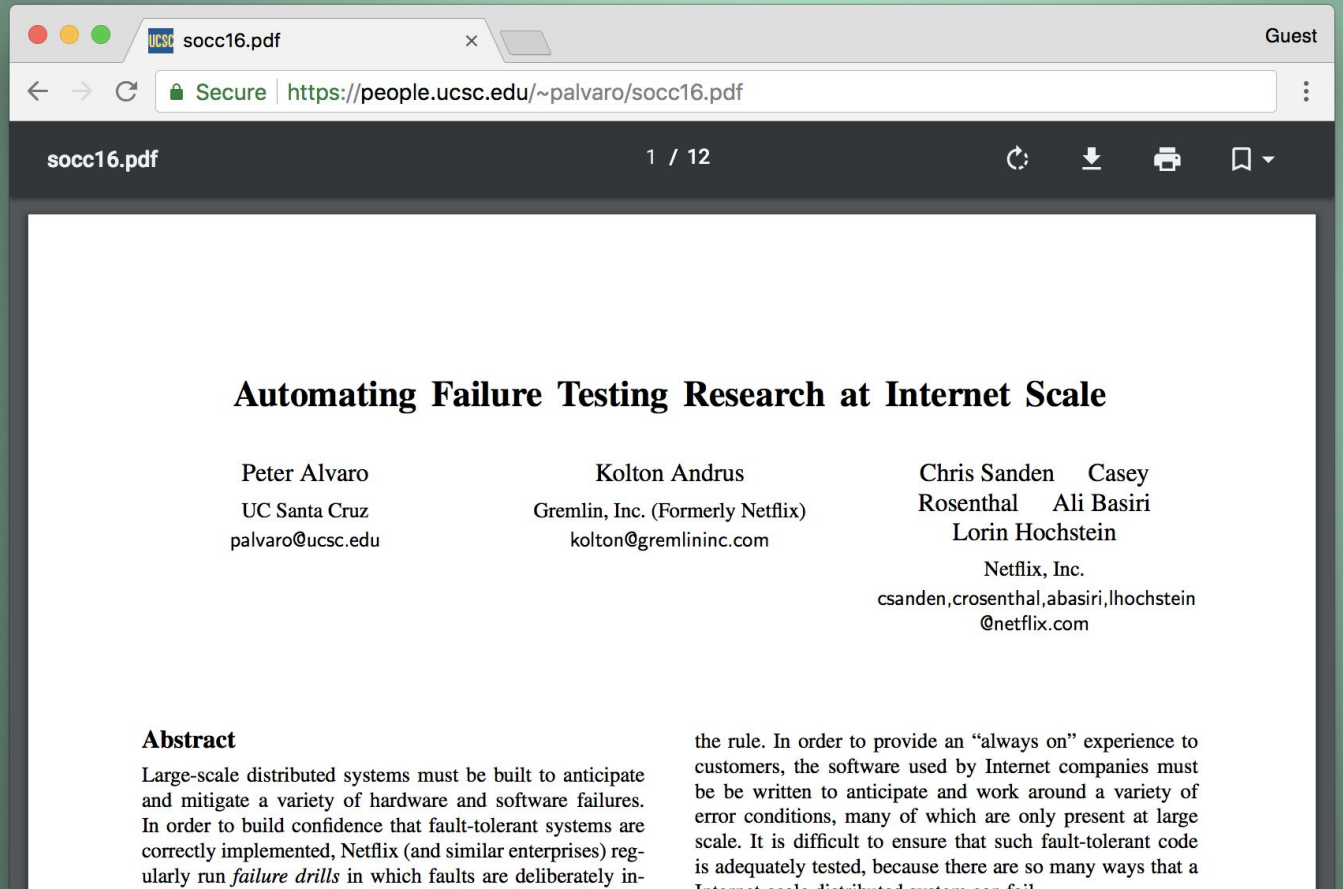
LABS

One of the advantages of being disorderly is that one is constantly making exciting discoveries. - A.A. Milne

Research

Distributed systems are ubiquitious, but they remain notoriously difficult to reason about and program. Our

CHAOS ENGINEERING BOOTCAMP



Automating Failure Testing Research at Internet Scale

Peter Alvaro
UC Santa Cruz
palvaro@ucsc.edu

Kolton Andrus
Gremlin, Inc. (Formerly Netflix)
kolton@gremlininc.com

Chris Sanden Casey Rosenthal Ali Basiri Lorin Hochstein
Netflix, Inc.
csanden,crosenthal,abasiri,lhochstein@netflix.com

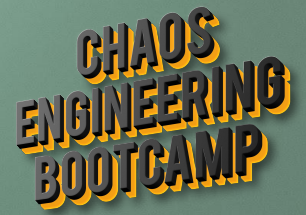
Abstract

Large-scale distributed systems must be built to anticipate and mitigate a variety of hardware and software failures. In order to build confidence that fault-tolerant systems are correctly implemented, Netflix (and similar enterprises) regularly run *failure drills* in which faults are deliberately in-

the rule. In order to provide an "always on" experience to customers, the software used by Internet companies must be be written to anticipate and work around a variety of error conditions, many of which are only present at large scale. It is difficult to ensure that such fault-tolerant code is adequately tested, because there are so many ways that a

Where to learn more?

Principles of Chaos
Chaos Engineering Book
Awesome Chaos Engineering GitHub Repo
Chaos Engineering Slack gremlin.com/slack
Gremlin Community
Netflix - Chaos Kong
Chaos Engineering Meetups
Learn More about SEV
Learn More about GameDays



Q&A

Any Questions?

THANKS!

we have stickers!

@ana_m_medina @HoReaL

CHAOS ENGINEERING BOOTCAMP