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.
Gremlin

Chaos Engineer /ˈkɑː,əs ˈenjəˈnɪr/ noun

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

gremlin.com
@gremlininc
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.
Join Slack

www.gremlin.com/slack
#srecon18_europe
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 1: 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*. 
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

cassandra

Amazon ECS

kafka

elasticsearch
Companies doing Chaos Engineering

Netflix
Twilio
Under Armour
Dropbox
Expedia
Uber
Jet
Amazon
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

- Sentry
- Honeycomb
- Datadog
Part 2: Breaking Things
Minimize the blast radius.

- Form a hypothesis.
- Run an experiment.
- Abort Conditions
- Failure
- Success
- Find and fix issues.
- Scale up and repeat.
Please get in groups of 3
Getting Access to your host:

While we wait, check out this survey:

bit.ly/ce-questions
ssh into your hosts as root

$ ssh root@207.154.216.247

Password: chaosbootcamp
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
gpg | 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
choose host that will be **master**

```bash
$ su -l kube
Password: chaosbootcamp
```

**initialize kubernetes master node**

```bash
$ sudo kubeadm init
```

**start cluster on master**

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

don master, verify nodes have joined master
$ sudo kubectl get nodes
cool! you’ve gotten Kubernetes set up, let’s setup the rest
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
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
On every host as root, let's get some monitoring in place with Datadog

```
$ DD_API_KEY=faff9c88d8cdd357d76505f595f23797
```

on every host, let's grab the bootcamp files

```
$ su - chaos
$ git clone https://github.com/tammybutow/chaosengineeringbootcamp
```
let the chaos begin
hello world, the chaos way
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
<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>SCR</th>
<th>%MEM</th>
<th>TIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>27878</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3580</td>
<td>3072 R</td>
<td>6.5</td>
<td>0.1</td>
<td>0.24</td>
<td>C008</td>
<td>openssl1</td>
</tr>
<tr>
<td>27879</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3525</td>
<td>3124 R</td>
<td>6.6</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27793</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3584</td>
<td>3072 R</td>
<td>6.3</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27795</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3592</td>
<td>3168 R</td>
<td>6.3</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27799</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3584</td>
<td>3124 R</td>
<td>6.3</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27811</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3618</td>
<td>3168 R</td>
<td>6.3</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27813</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3618</td>
<td>3032 R</td>
<td>6.3</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27834</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3522</td>
<td>3124 R</td>
<td>6.3</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27835</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3600</td>
<td>3168 R</td>
<td>6.3</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27836</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3522</td>
<td>3216 R</td>
<td>6.3</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27797</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3168 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27798</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27837</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3168 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27804</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27752</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27753</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27757</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27758</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3168 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27762</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3168 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27766</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27768</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27770</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27771</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27772</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
<tr>
<td>27773</td>
<td>chaos</td>
<td>20</td>
<td>0</td>
<td>13532</td>
<td>3532</td>
<td>3032 R</td>
<td>5.9</td>
<td>0.1</td>
<td>0.24</td>
<td>C004</td>
<td>openssl1</td>
</tr>
</tbody>
</table>
CPU usage (%)

- 08:00 to 11:00
- CPU usage: 50% - 100%
lets stop the known chaos

$ pkill -u chaos

$ top
Part 3: 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 applause
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 4:
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
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
Why do Chaos Engineering?
prevent outages
Northeast blackout of 2003

From Wikipedia, the free encyclopedia

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

* This article has an unclear 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 dropped 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.
Outage post-mortem

Akhil Gupta | January 12, 2014

On Friday evening our service went down during scheduled maintenance. The service was back up and running about three hours later, with core service fully restored by 4:40 PM PT on Sunday.

For the past couple of days, we’ve been working around the clock to restore full access as soon as possible. Though we’ve shared some brief updates along the way, we owe you a detailed explanation of what happened and steps we are taking to ensure it doesn’t happen again.
Some master-replica pairs were impacted which resulted in site going down

Scaling Uber with Matt Ranney

By Pranay

Podcast | Friday, December 4 2015
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 5: 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 6:
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
Failure Fridays

Dogfooding.

We use Gremlin for Chaos Engineering Experiments

Dedicated time to practice Chaos Engineering to reveal weaknesses in our services
Part 7: Advanced Chaos
Other Uses for Chaos Engineering
Preparation for IPO / Auditing
Training new engineers / on-call practicing
Testing alert thresholds
Research

Distributed systems are ubiquitous, but they remain notoriously difficult to reason about and program. Our research at Disorderly Labs centers on the intersection of distributed systems and language support.
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  
Ali Basiri  
Lorin Hochstein  
Netflix, Inc.

csanden,crossenthal,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 introduced to test the system’s ability to recover from these events. This technique is essential to ensure that the systems remain operational despite such failures. However, not all faults can be accurately mimicked, and the costs of simulating all possible failure conditions are prohibitive. The ability to execute these failure drills at scale is limited, and improving the efficiency of these tests can lead to reductions in costs and improvements in system reliability.

In this paper, we present an automated framework that integrates failure testing at Internet scale. Our approach leverages machine learning techniques to predict and induce failure conditions, thus automating the failure testing process. By allowing for the efficient execution of failure drills, our solution provides a more effective means of testing system resilience, ultimately leading to improved system reliability and customer experience.
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