
AWS
re:Invent

OPN304

Open-source failure injection on AWS

Adrian Hornsby
Principal Developer Advocate
AWS

Varun Jewalikar
Software Engineer
Amazon Prime Video

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“Failures are a given, and everything will eventually fail over time.”

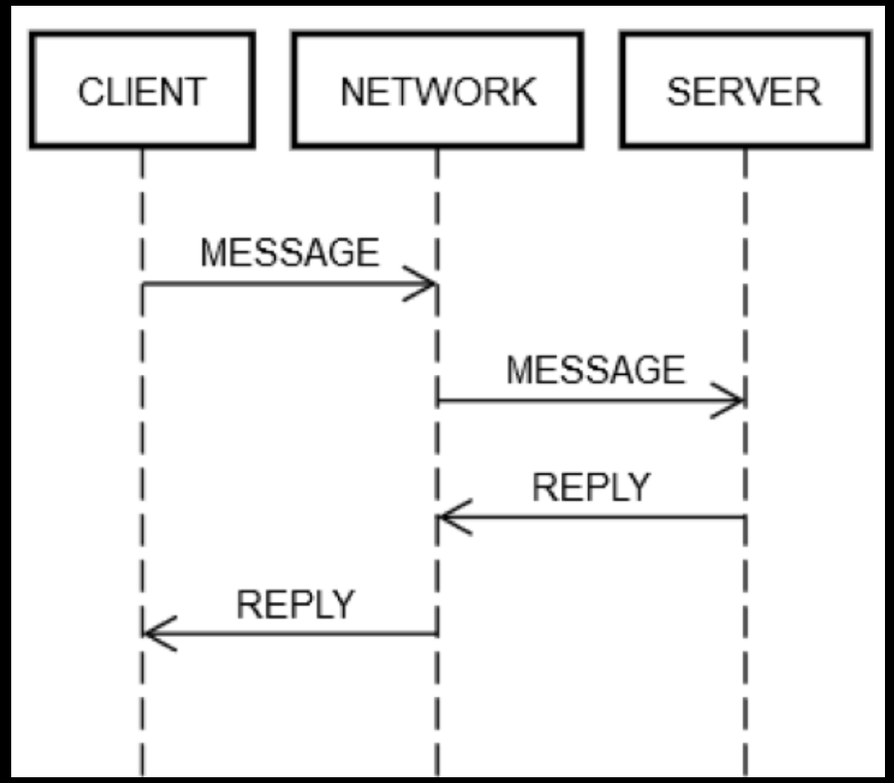
Werner Vogels

CTO

Amazon.com



What could possibly go wrong?



https://aws.amazon.com/builders-library/challenges-with-distributed-systems/

Is traditional testing enough? NO

Testing – verifying a **known** condition

E.g., assert(A = B) ?

Unit testing of components

Tested in isolation to ensure function meets expectations

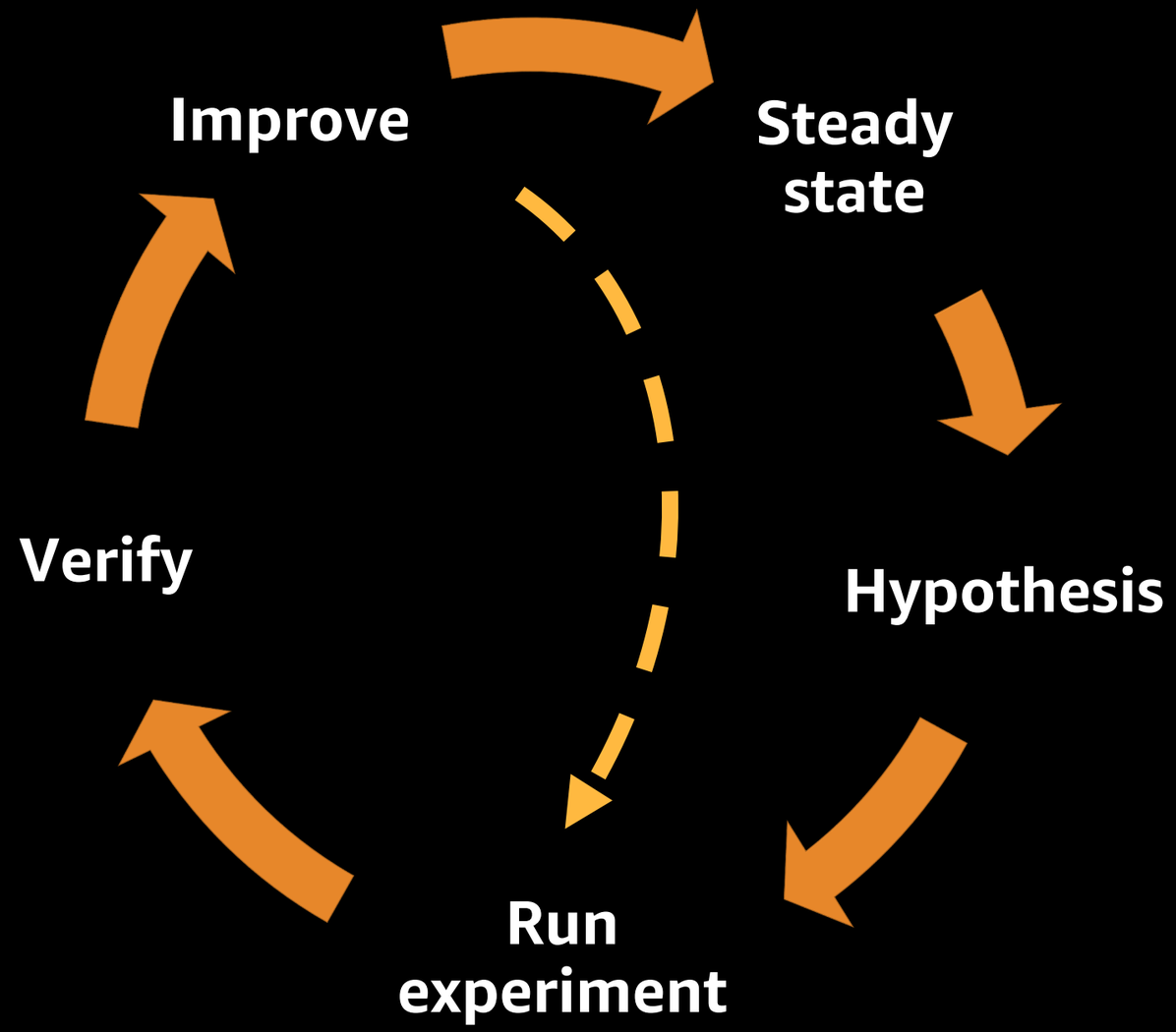
Functional testing of integrations

Each execution path tested to assure expected results



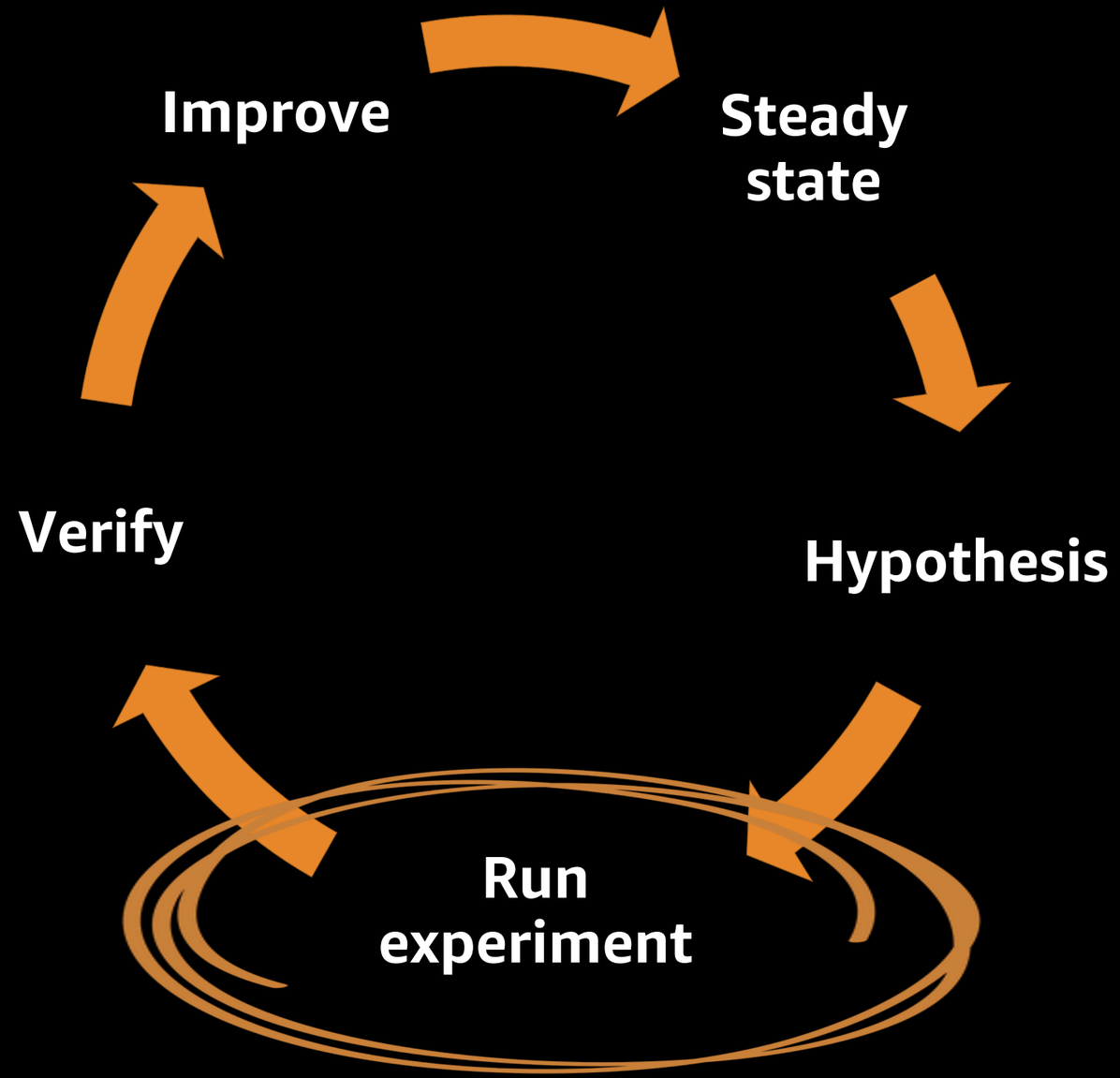
Chaos engineering

A SCIENTIFIC METHOD



- ✓ Building confidence against failures
- ✓ Validates monitoring, observability & alarms
- ✓ Reducing skill atrophy
- ✓ Improving recovery time
- ✓ And a lot more

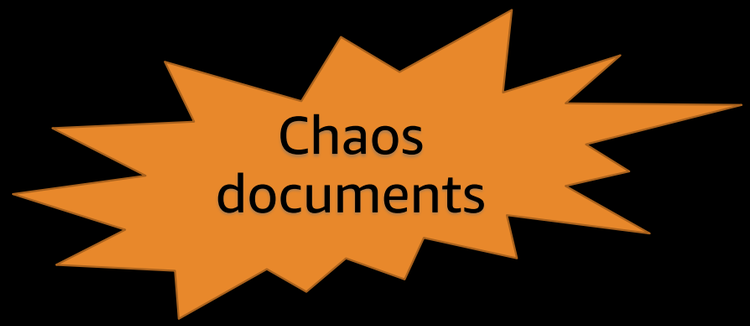
Phases of chaos engineering



Failure injection

- Application-level failure injection (throw exceptions, errors, etc.)
- Host level (kill services, processes, etc.)
- Resource exhaustion (exhaust CPU, memory, IO, etc.)
- Network failure (inject latency, packet loss, etc.)
- Subnet failure (deny routing policy)
- Service failure (force DB reboot, failover, etc.)
- Organization (people, processes, etc.)

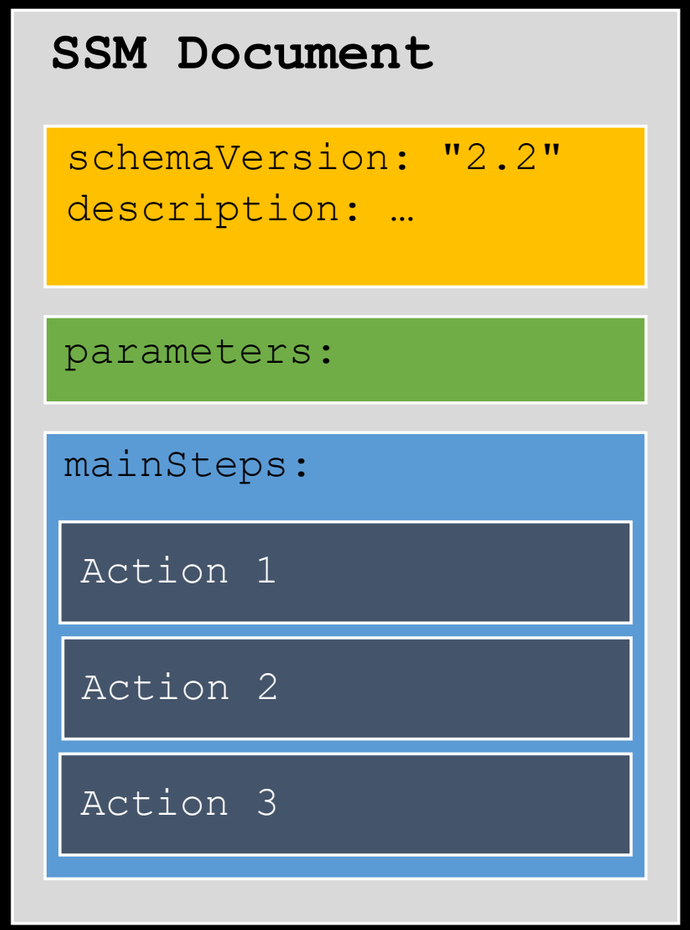






Failure injection
“chaos” documents

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



chaos-ssm-documents > run-command > ! cpu-stress.yml > ...

```yaml
---
description: Run a CPU stress on an instance
schemaVersion: '2.2'
parameters:
  duration:
    type: String
    description: The duration - in seconds - of the attack. (Required)
    default: "60"
  cpu:
    type: String
    description: 'Specify the number of CPU stressors to use (default all)'
    default: "0"
mainSteps:
- action: aws:runShellScript
  name: ExecuteStressNg
  inputs:
    runCommand:
    # https://www.mankier.com/1/stress-ng#Examples
    - "stress-ng --cpu {{ cpu }} --cpu-method matrixprod -t {{ duration }}s"
```

```
mainSteps:
- action: aws:runShellScript
  name: RevertCommandScheduling
  inputs:
    runCommand:
    - "echo 'tc filter del dev eth0 prio 1' | at now + {{ duration }} minutes"
    - "echo 'tc qdisc del dev eth0 parent 1:1 handle 10:' | at now + {{ duration }} minutes"
    - "echo 'tc qdisc del dev eth0 root handle 1: prio' | at now + {{ duration }} minutes"
- action: aws:runShellScript
  name: AlterNetworkInterface
  inputs:
    runCommand:
    - export REGION=`curl http://169.254.169.254/latest/dynamic/instance-identity/document|grep region|awk -F\" '{print $4}'`
    - "tc qdisc add dev eth0 root handle 1: prio priomap 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2"
    - "tc qdisc add dev eth0 parent 1:1 handle 10: netem delay {{ delay }}ms {{ jitter }}ms distribution normal"
    - "for k in $(curl -s https://ip-ranges.amazonaws.com/ip-ranges.json | jq -r '.prefixes[] | select(.region==env.REGION) |
    select(.service==\"{{ service }}\") | .ip_prefix'); do echo $k && tc filter add dev eth0 protocol ip parent 1:0 prio 1
    u32 match ip dst $k match ip dport 443 0xffff flowid 1:1; done"
    - "tc qdisc show"
```



adhorn / **chaos-ssm-documents**

Unwatch

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights

Collection of AWS SSM Documents to perform Chaos Engineering experiments

chaos-engineering | chaos-monkey | chaos-testing | aws | aws-ec2 | amazon-web-services | software-engine

15 commits | **1** branch | **0** releases | **1** contributo

Branch: **master** | **New pull request** | **Create new file** | Upload f

adhorn improve readme

LICENCE	Adding Licence
README.rst	improve readme
blackhole-dns-stress.yml	Adding more attacks and fixing some typos
blackhole-dynamo-stress.yml	Adding dynanodb blackhole attack
blackhole-ec2-stress.yml	Adding EC2 blackhole attack
blackhole-s3-stress.yml	Adding S3 blackhole attack
blackhole-stress.yml	Adding more attacks and fixing some typos
cpu-stress.yml	Adding more attacks and fixing some typos
io-stress.yml	Adding more attacks and fixing some typos
latency-delta-stress.yml	Adding more attacks and fixing some typos
latency-stress.yml	Adding more attacks and fixing some typos
memory-stress.yml	Adding more attacks and fixing some typos
network-corruption-stress.yml	Adding more attacks and fixing some typos
network-loss-stress.yml	Adding more attacks and fixing some typos
upload-document.sh	adding tags

Windows support

adhorn / chaos-ssm-documents

Unwatch 13 | Star 151 | Fork 39

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | ...

master | chaos-ssm-documents / run-command / windows / | Go to file | Add file



thebeebs Adding Windows Documents ... 12 hours ago History

..

README.MD	Adding Windows Documents	12 hours ago
blackhole-dns-stress.yml	Adding Windows Documents	12 hours ago
blackhole-dynamo-stress.yml	Adding Windows Documents	12 hours ago
blackhole-ec2-stress.yml	Adding Windows Documents	12 hours ago
blackhole-rds-stress.yml	Adding Windows Documents	12 hours ago
blackhole-s3-stress.yml	Adding Windows Documents	12 hours ago
blackhole-stress.yml	Adding Windows Documents	12 hours ago
cpu-stress.yml	Adding Windows Documents	12 hours ago

README.MD

Experimental

The Windows Run Commands are all currently experimental and should not be used until we have tested more

Currently available

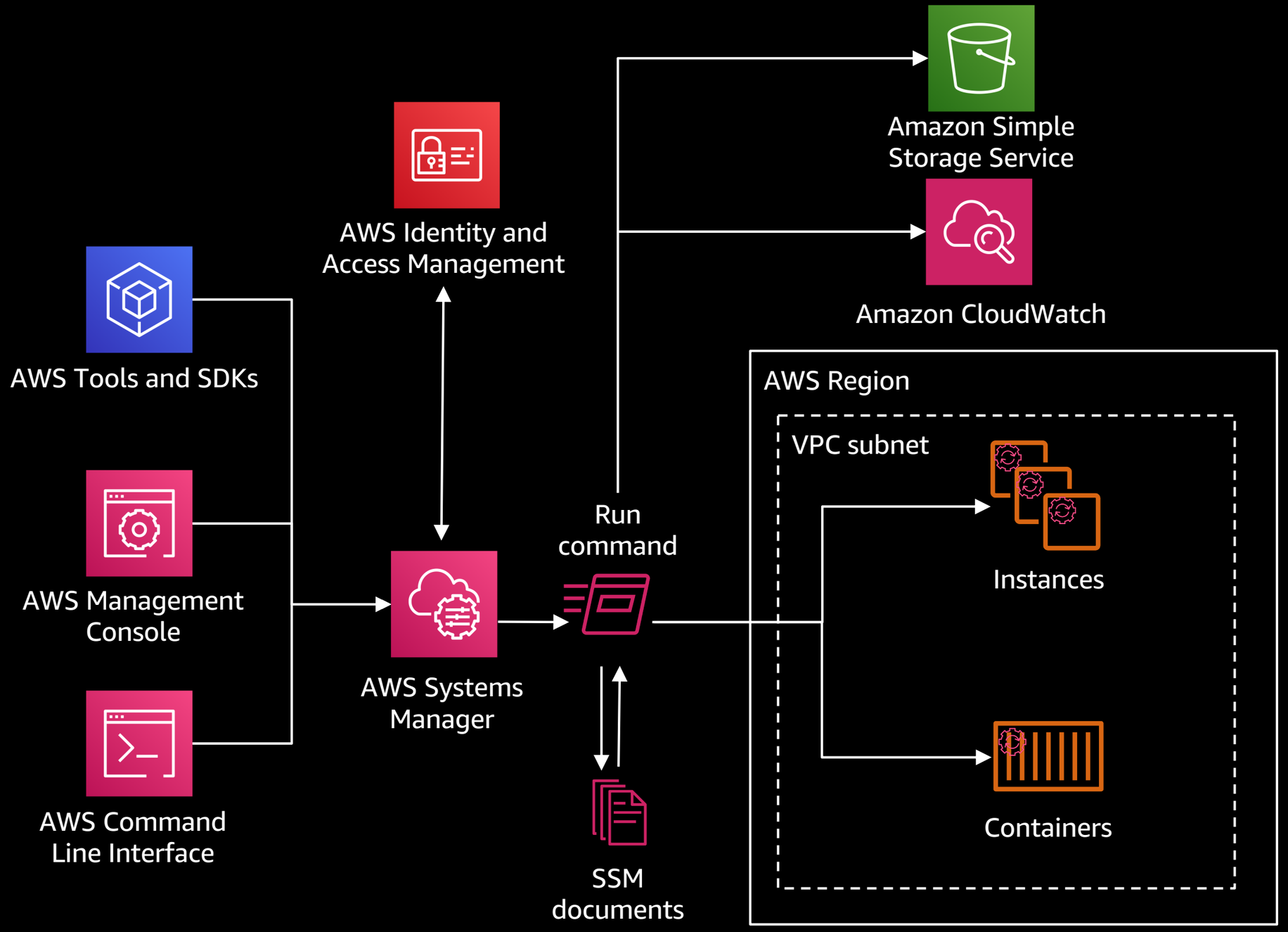
- Latency injection (for all traffic or selected AWS services)
- Blackholing
- CPU, memory, and IO hogging
- Fill up disk
- Network packet loss (using traffic shaping)
- Network packet corruption (using traffic shaping)
- Kills random processes, services, and instances
- Detach (force) Amazon EBS volumes
- Alter /etc/hosts
- And more.

Running “chaos” documents with AWS Systems Manager



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.


Amazon Simple Storage Service
AWS Identity and Access Management
Amazon CloudWatch
AWS Tools and SDKs
AWS Region
VPC subnet
Run command
AWS Management Console
Instances
AWS Systems Manager
AWS Command Line Interface
Containers
SSM documents

AWS Systems Manager Agent (SSM Agent)



Search or jump to... / Pull requests Issues Marketplace Explore

aws / amazon-ssm-agent — Watch 93 | Star 620 | Fork 189

Code | Issues 71 | Pull requests 7 | Actions | Projects | Security | Insights

master | 1 branch | 96 tags | Go to file | Add file | Code

Updated release notes ... 5061338 16 hours ago 2,059 commits

Tools/src	Fixing travis build	16 hours ago
agent	Support TCP multiplexing in port plugin	16 hours ago
common	Squash commit with changes for SSM Agent V3	last month
core	Added agent and worker version logging	16 hours ago
internal/tests	Squash commit with changes for SSM Agent V3	last month
packaging	Update systemd configuration to always restart the agent w...	16 hours ago
src	add back src folder	5 years ago
vendor/src	Add smux library for TCP multiplexing	16 hours ago
.gitignore	Updated the darwin patch set to work with the newer versio...	2 years ago
.travis.yml	Fixing travis build	16 hours ago
CODE_OF_CONDUCT.md	Create CODE_OF_CONDUCT.md	3 years ago
CONTRIBUTING.md	Updating GoLang version in travis file and updating the dev...	3 years ago
Config	Squash commit with changes for SSM Agent V3	last month
LICENSE	Update LICENSE	3 years ago
NOTICE.md	Add smux library for TCP multiplexing	16 hours ago
README.md	Add new config parameters to README.md	16 hours ago
RELEASENOTES.md	Updated release notes	16 hours ago

About

Agent to enable remote management of your Amazon EC2 instance configuration.

aws.amazon.com/ec2/run-com...

Readme

Apache-2.0 License

Releases 96

Amazon SSM Agent - ... Latest
16 hours ago

+ 95 releases

Packages

No packages published

Contributors 50

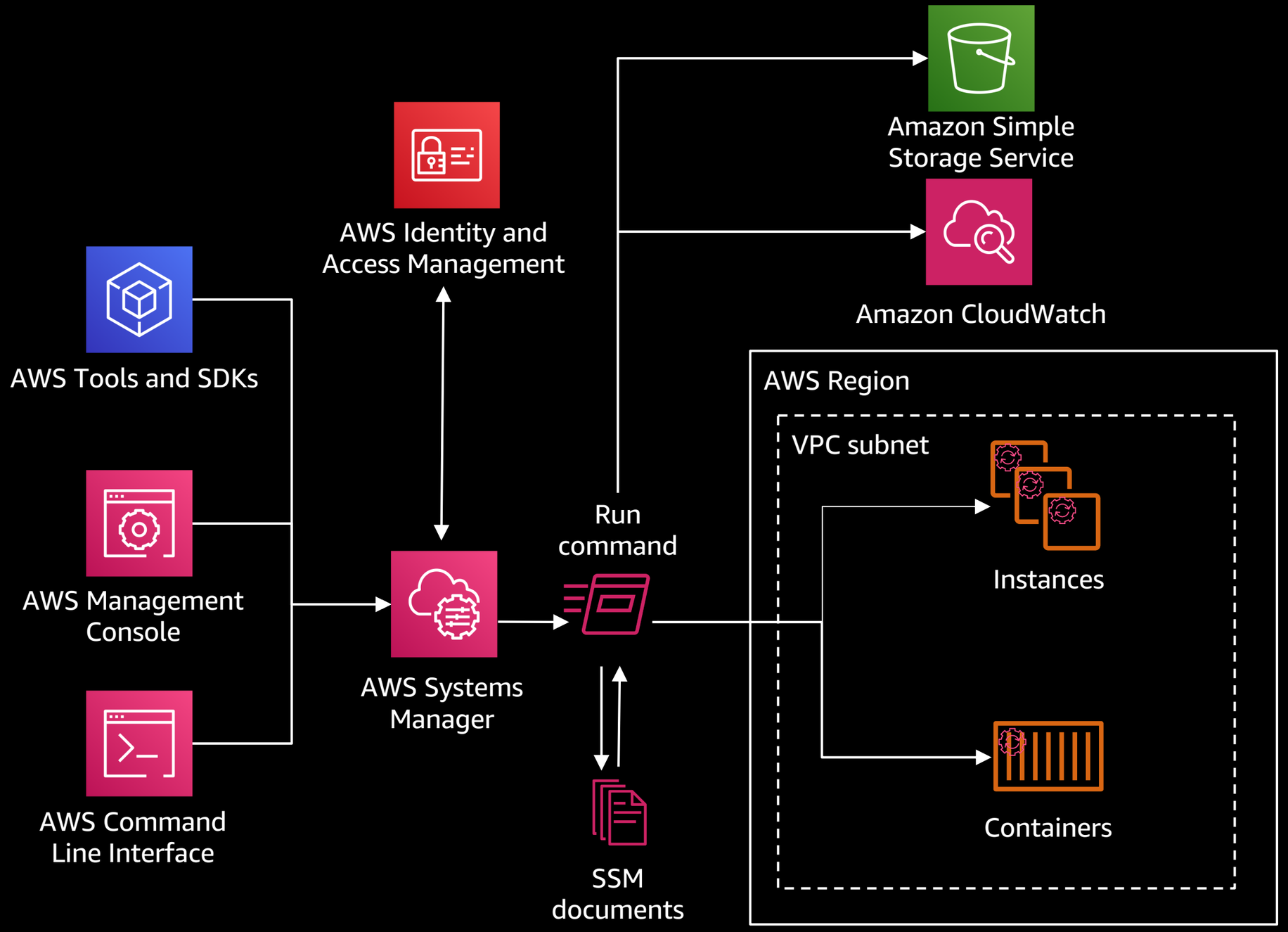
+ 39 contributors

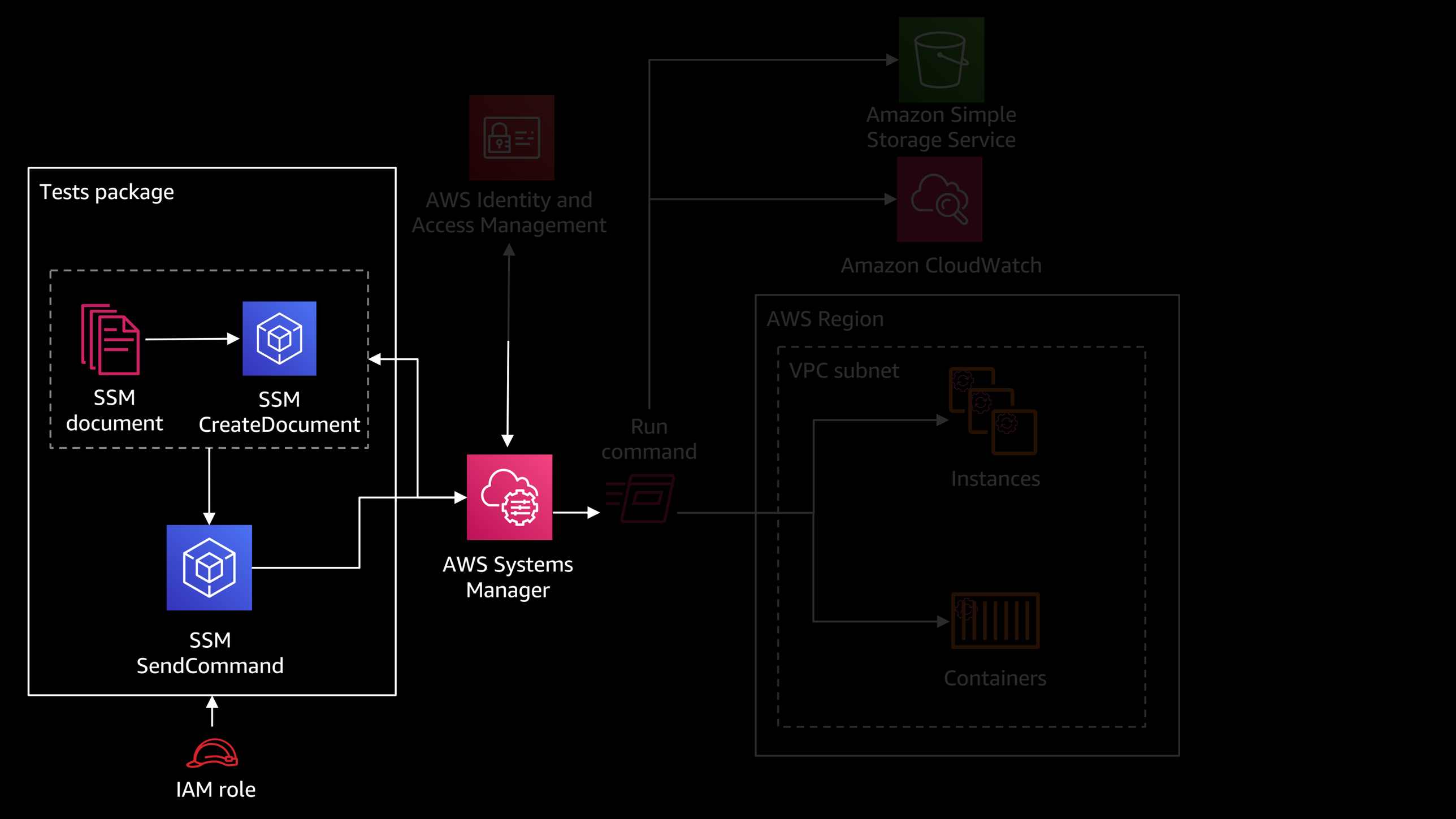
AWSSSMChaosRunner library

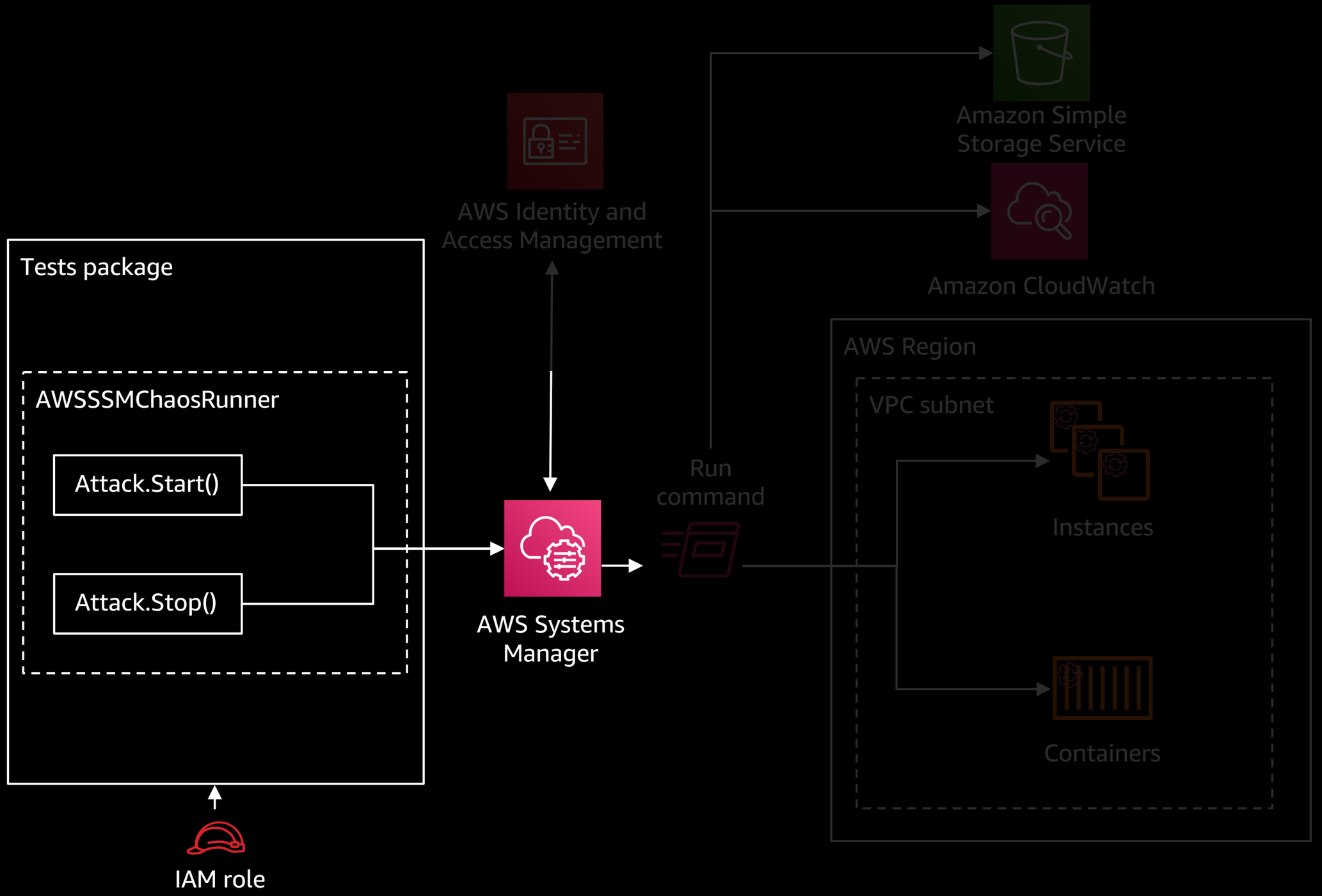


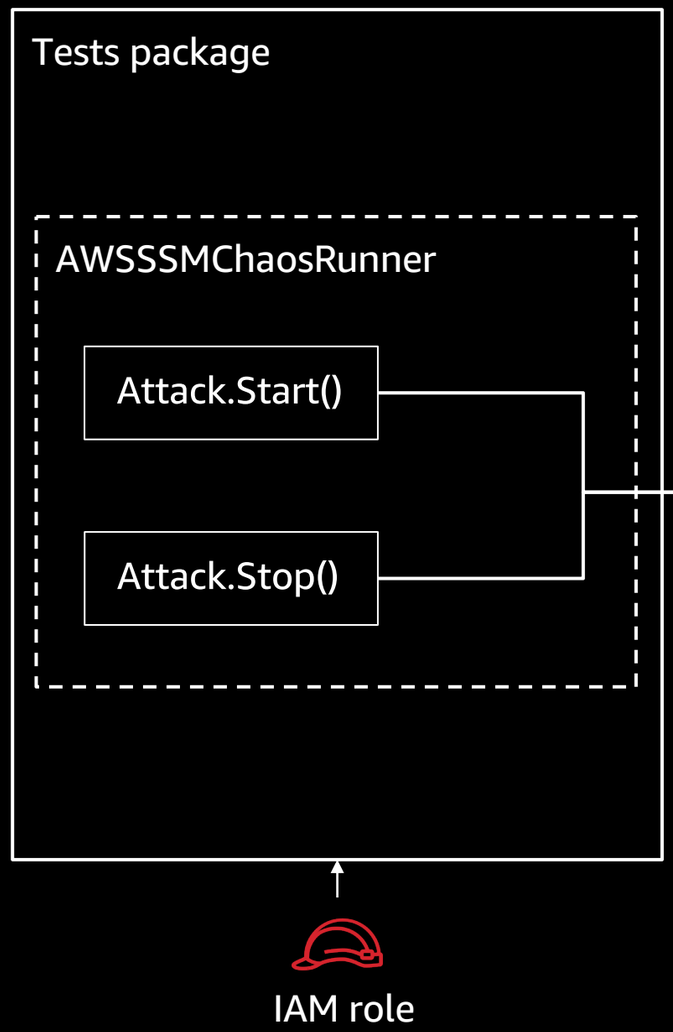
AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.


Amazon Simple Storage Service
AWS Identity and Access Management
Amazon CloudWatch
AWS Tools and SDKs
AWS Region
VPC subnet
Run command
AWS Management Console
Instances
AWS Systems Manager
AWS Command Line Interface
Containers
SSM documents


Tests package
SSM document
SSM CreateDocument
SSM SendCommand
IAM role
AWS Identity and Access Management
AWS Systems Manager
Run command
Amazon Simple Storage Service
Amazon CloudWatch
AWS Region
VPC subnet
Instances
Containers


Amazon Simple Storage Service
AWS Identity and Access Management
Amazon CloudWatch
Tests package
AWS Region
AWSSSMChaosRunner
VPC subnet
Attack.Start()
Run command
Instances
Attack.Stop()
AWS Systems Manager
Containers
IAM role



Amazon Simple Storage Service

AWS Identity and Access Ma

AWS Sy Mana

```
import software.amazon.awssmchaosrunner.attacks.SSMAttack
import software.amazon.awssmchaosrunner.attacks.SSMAttack.Companion.getAttack
...

@Before
override fun initialise(args: Array<String>) {
    if (shouldExecuteChaosRunner()) {
        ssm = applicationContext.getBean(AWSSimpleSystemsManagement::class.java)
        ssmAttack = getAttack(ssm, attackConfiguration)
        command = ssmAttack.start()
    }
}

@After
override fun destroy() {
    ssmAttack.stop(command)
}
```

- ✓ Abstracts Systems Manager details
- ✓ Tried and tested failure injections
- ✓ Tested Amazon EC2 and Amazon ECS (with EC2 launch type)
- ✓ Active community and support
- ✓ And a lot more



amzn / awsssmchaosrunner

Unwatch 13 | Unstar 164 | Fork 6

Code | Issues | Pull requests | Actions | Security | Insights | Settings

master | Go to file | Add file | Code

neo01124 Update version ... ✓ on 21 Aug 50

gradle/wrapper	Add gradle	4 months ago
src	Merge mainline and fix null in AbstractDepen...	last month
.gitignore	Add gradle	4 months ago
.travis.yml	Add badges	3 months ago
AWSSSMChaosRunn...	Add README and LICENSE	6 months ago
CODE_OF_CONDUCT...	Initial commit	4 months ago
CONTRIBUTING.md	Initial commit	4 months ago
LICENSE	Initial commit	4 months ago
NOTICE	Add NOTICE & THIRD-PARTY	4 months ago
README.md	Update version	last month
THIRD-PARTY	Add NOTICE & THIRD-PARTY	4 months ago
build.gradle	Update version	last month
gradle.properties	Improve build.gradle and add maven publishi...	3 months ago
gradlew	Remove internal tooling	4 months ago
gradlew.bat	Remove internal tooling	4 months ago

README.md

build passing | maven central 1.3.0 | javadoc 1.3.0

About

Amazon's light-weight library for chaos engineering on AWS. It can be used for EC2, ECS (with EC2 launch type) and AWS Fargate.

ssm | aws-ssm | ec2 | chaos-engineering | fargate | ecs | failure-injection | aws

Readme

Apache-2.0 License

Contributors 4

neo01124 Varun Jewali...

amazon-auto Amazon ...

Languages

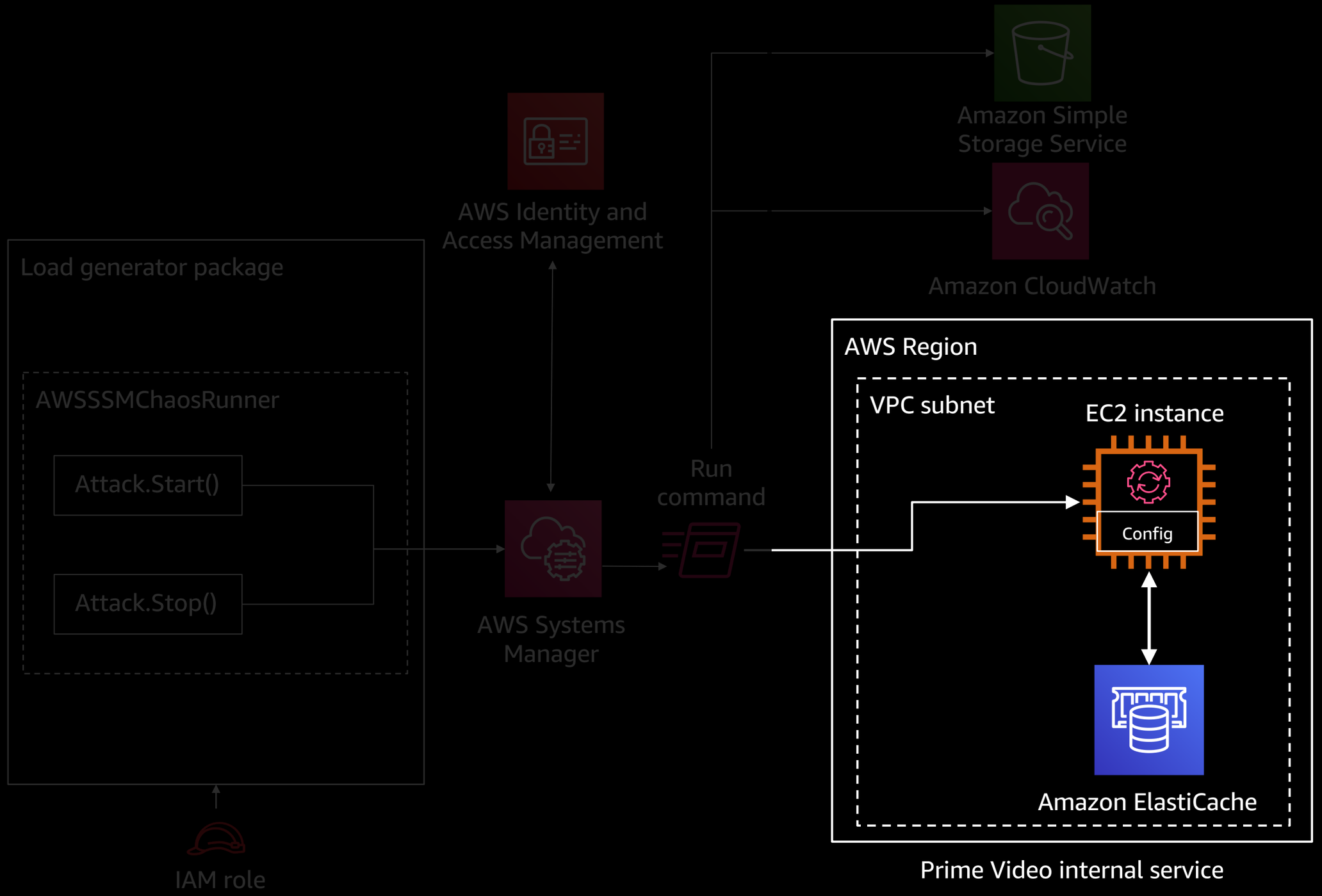
Kotlin 100.0%

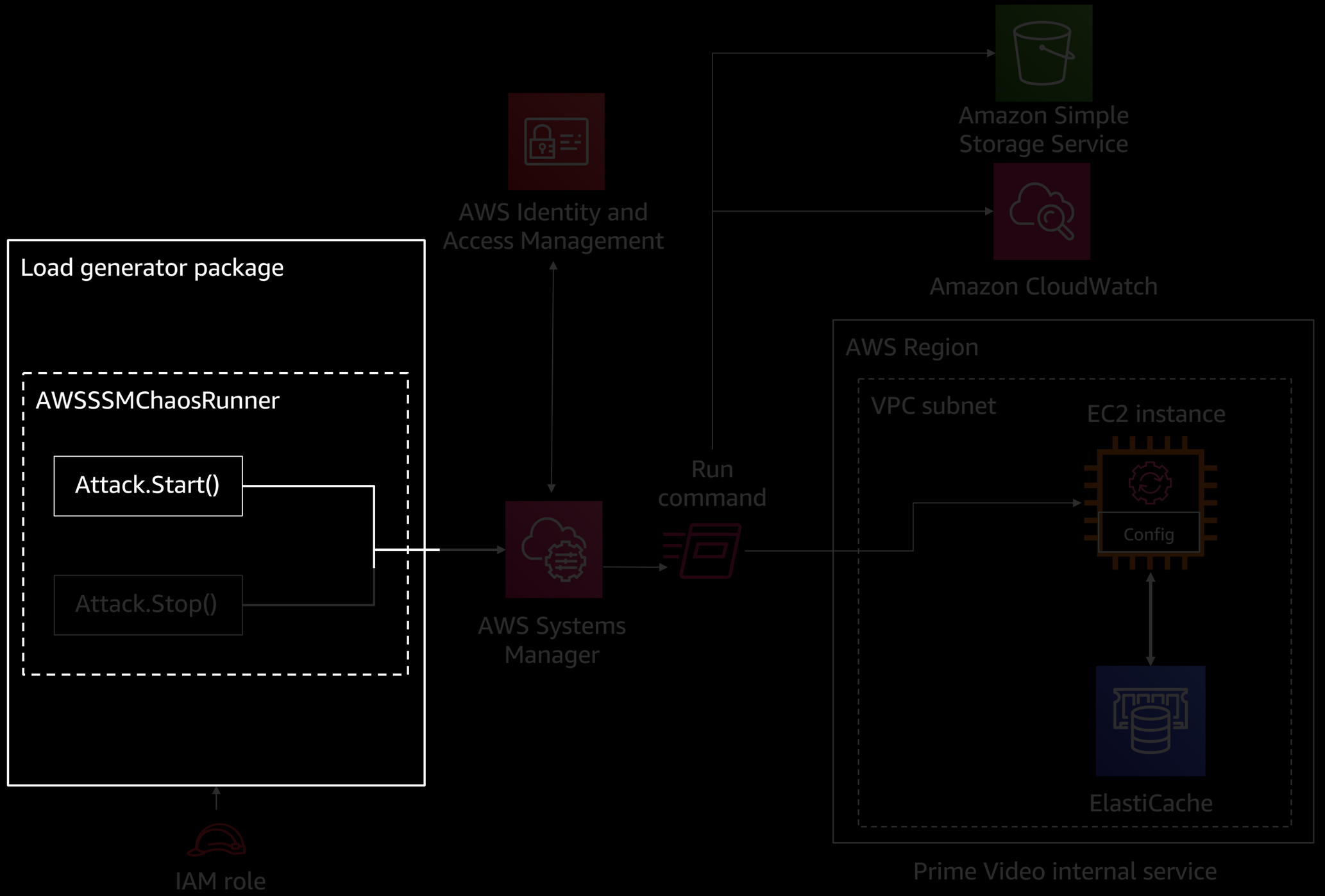
Amazon Prime Video prevents an outage

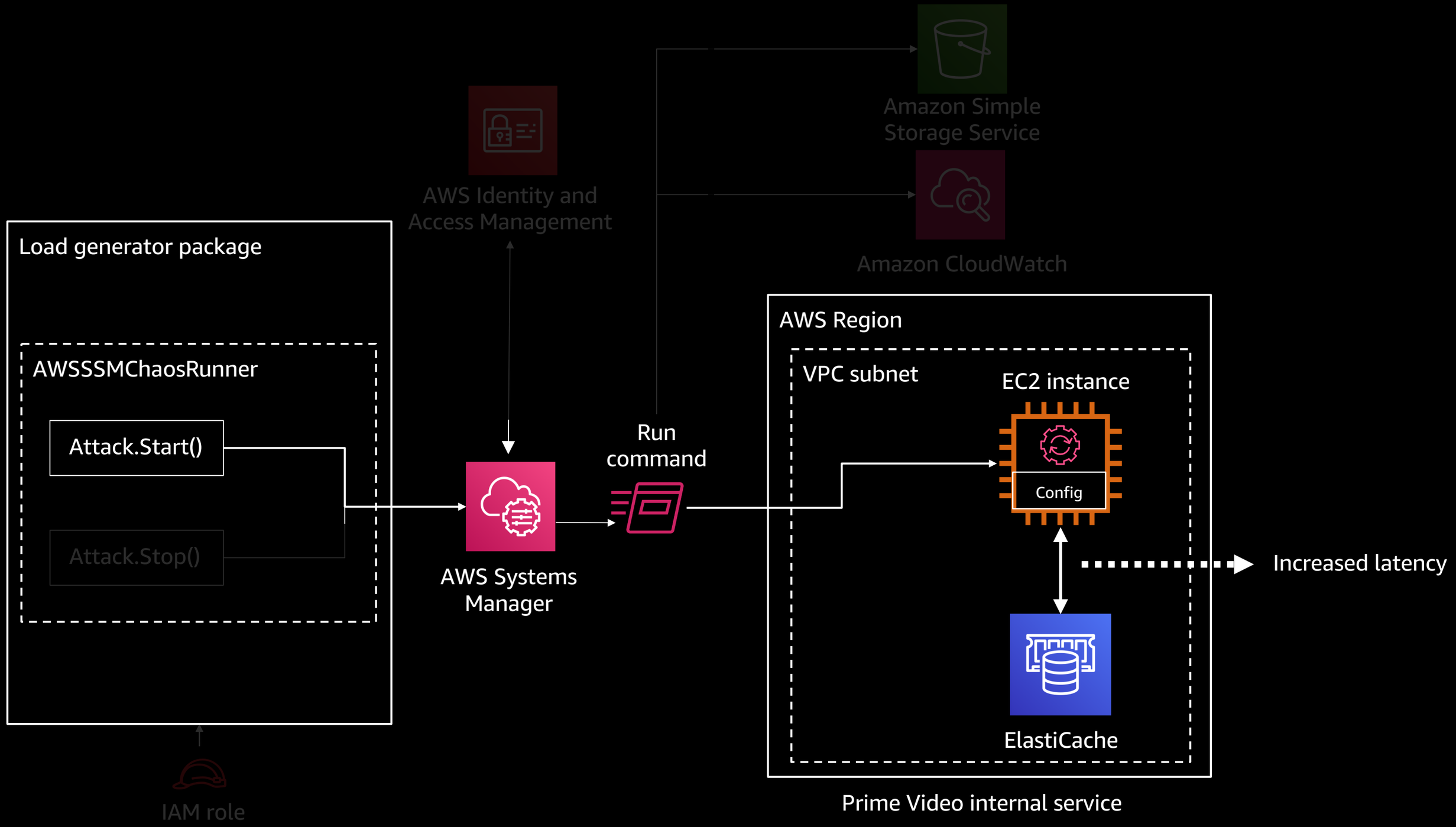


AWS re:Invent

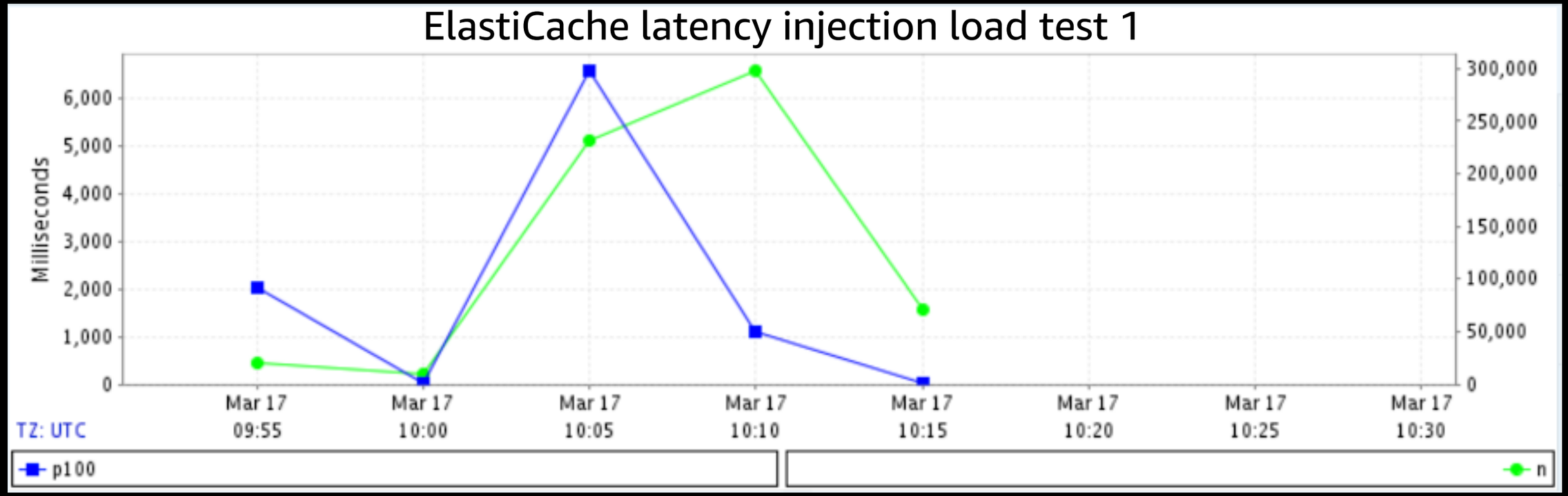
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.


Amazon Simple Storage Service
AWS Identity and Access Management
Amazon CloudWatch
Load generator package
AWS Region
AWSSSMChaosRunner
VPC subnet
EC2 instance
Attack.Start()
Run command
Config
Attack.Stop()
AWS Systems Manager
Amazon ElastiCache
IAM role
Prime Video internal service

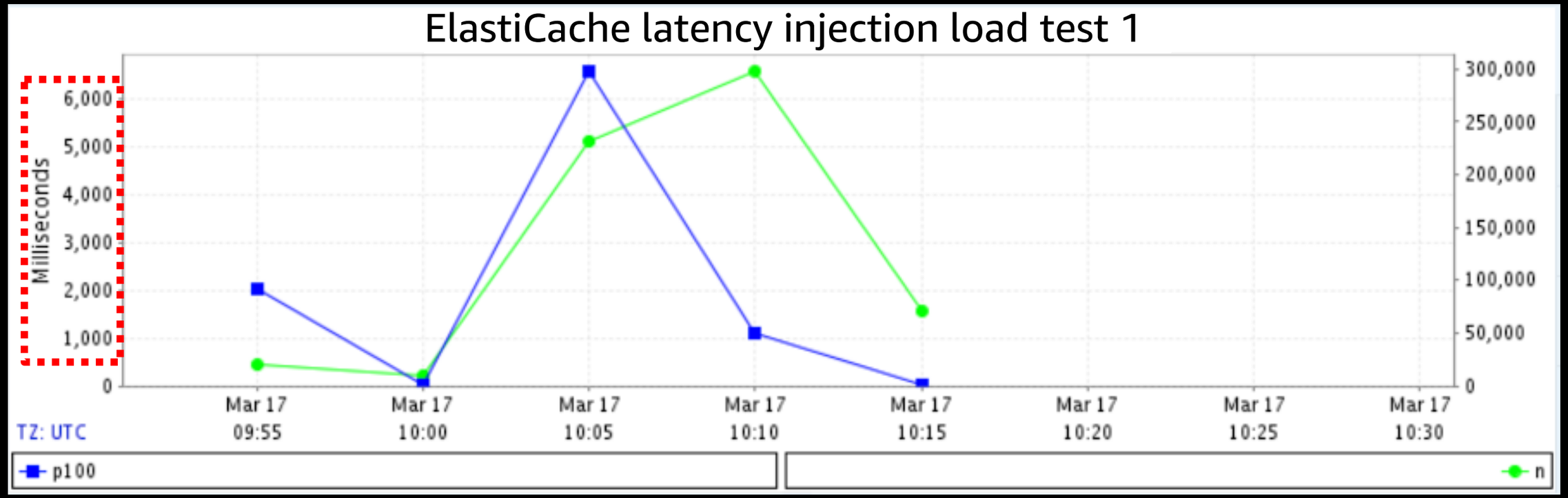

Amazon Simple Storage Service
AWS Identity and Access Management
Amazon CloudWatch
Load generator package
AWS Region
AWSSSMChaosRunner
VPC subnet
EC2 instance
Attack.Start()
Run command
Config
Attack.Stop()
AWS Systems Manager
ElastiCache
IAM role
Prime Video internal service


Amazon Simple Storage Service
AWS Identity and Access Management
Amazon CloudWatch
Load generator package
AWS Region
AWSSSMChaosRunner
VPC subnet
EC2 instance
Attack.Start()
Run command
Config
Attack.Stop()
AWS Systems Manager
Increased latency
ElastiCache
IAM role
Prime Video internal service

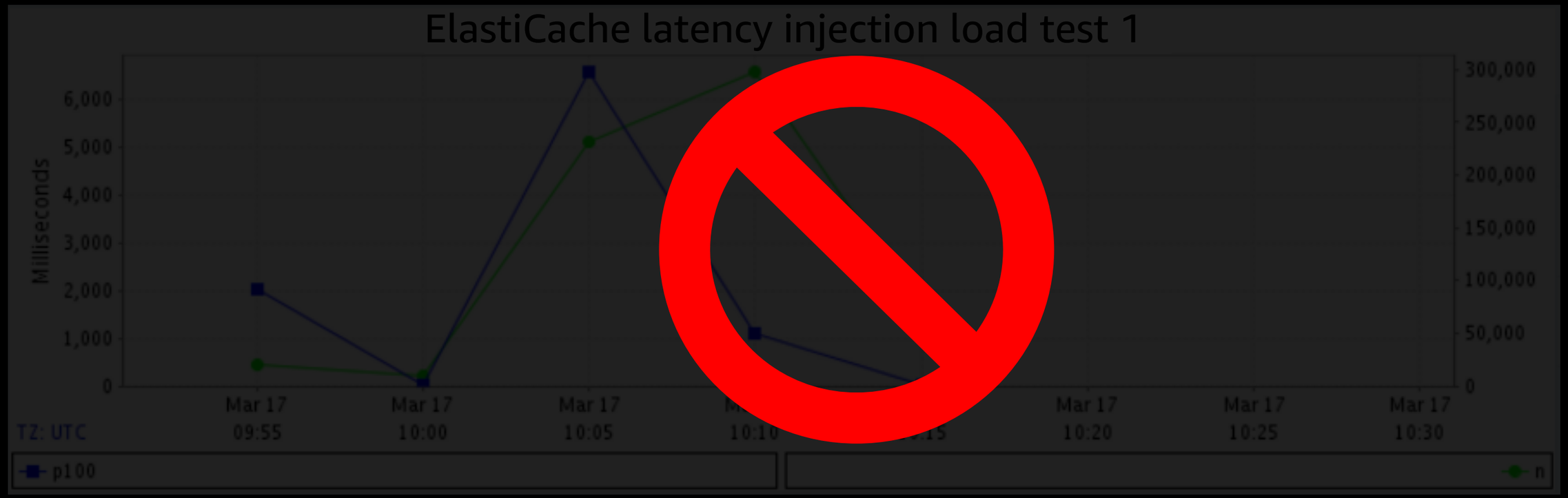
Amazon ElastiCache experiment #1



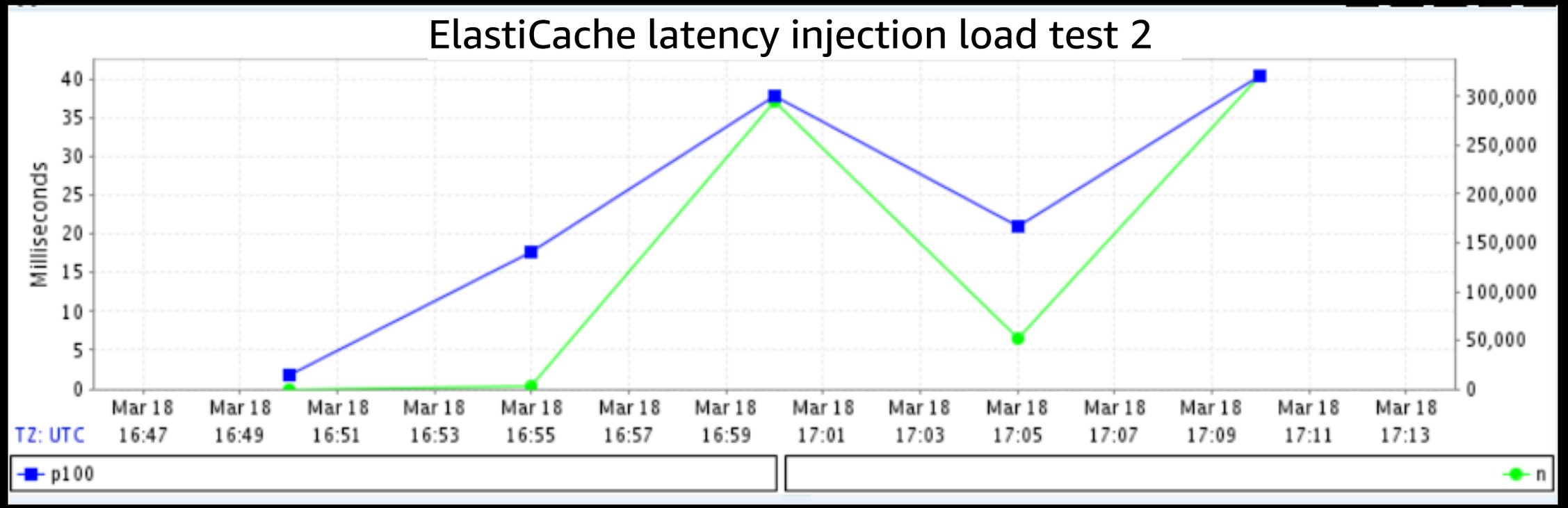
ElastiCache experiment #1



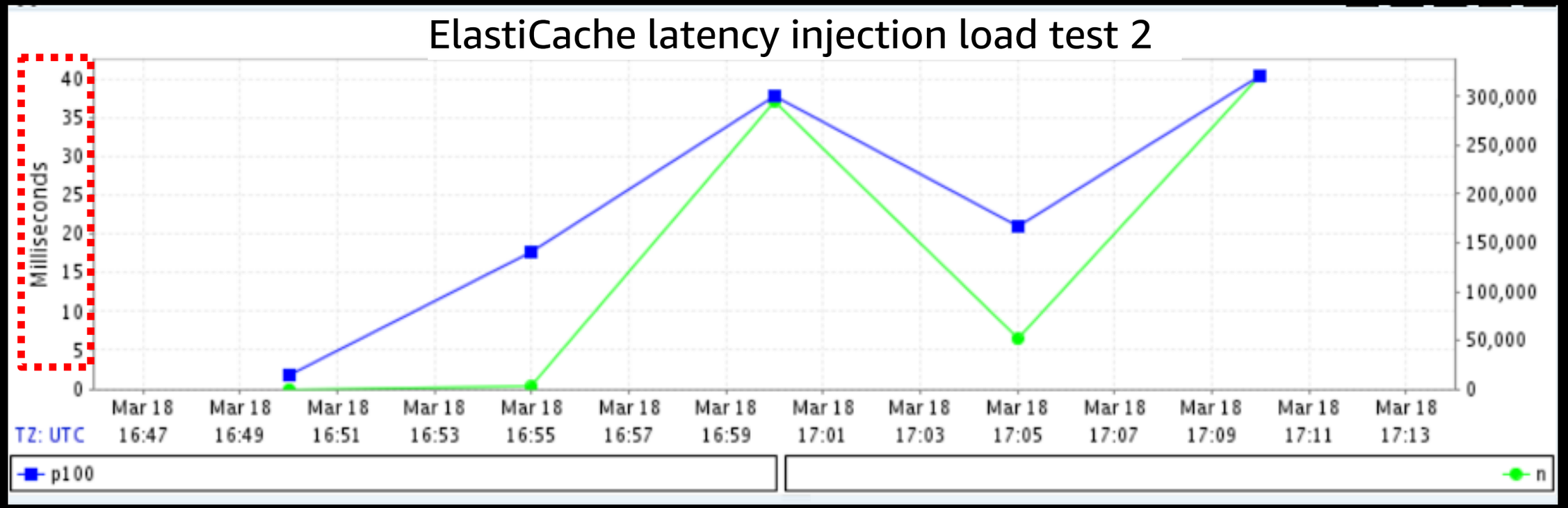
ElastiCache experiment #1



ElastiCache experiment #2



ElastiCache experiment #2



Help us improve the tools

- Join the conversations
- Report issues
- Contribute



AWS Open Source Blog

Building resilient services at Prime Video with chaos engineering

by Varun Jewalikar and Adrian Hornsby | on 18 AUG 2020 | in Amazon EC2, AWS Systems Manager, Open Source, Technical How-to | Permalink | Comments | Share

Large-scale distributed software systems are composed of several individual sub-systems—such as CDNs, load balancers, and databases—and their interactions. These interactions sometimes have unpredictable outcomes caused by unforeseen turbulent events (for example, a network failure). These events can lead to system-wide failures.

Chaos engineering is the discipline of experimenting on a distributed system to build confidence in the system's capability to withstand turbulent events. Chaos engineering requires adopting practices to identify interactions in distributed systems and related failures proactively, and also needs implementing and validating countermeasures. The key to chaos engineering is injecting failure in a controlled manner.

In this post, we present a simple approach for fault injection in systems utilizing Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Elastic Container Service (Amazon ECS), and its integration with a load-testing suite to validate the countermeasures put in place to prevent dependency and resource exhaustion failures. A typical chaos experiment could be generating baseline load (traffic) against the system, adding latency to all network calls to the underlying database, and then validating timeouts and retries. We will explain how to inject such failure (addition of latency to database calls), why validating countermeasures (timeouts and retries) under load is essential, and how to execute it in an Amazon EC2-based system.

We will start with a brief introduction to chaos engineering, then dive deep into failure injection using the AWS Systems Manager. We will then present our open source library, AWSSSMChaosRunner. This was inspired by Adrian Hornsby's "Injecting Chaos to Amazon EC2 using AWS Systems Manager" blog post.

Finally, we will provide an example of integration and explain how Prime Video used this library to prevent potentially customer-impacting outages.

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.