splunk> .conf2017

Dockerizing Splunk At Scale

Brian Bingham | Software Engineer

Brent Boe | Software Engineer

September 2017 | Washington, DC

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

splunk> .conf2017

What's Splunk At Scale?

- How about 20 TB / day ingestion and event generation
  - 100 Beefy Servers
  - 20 Week Run-time
  - 7 Engineers
  - Multiple Installed Apps
  - Datamodel Acceleration

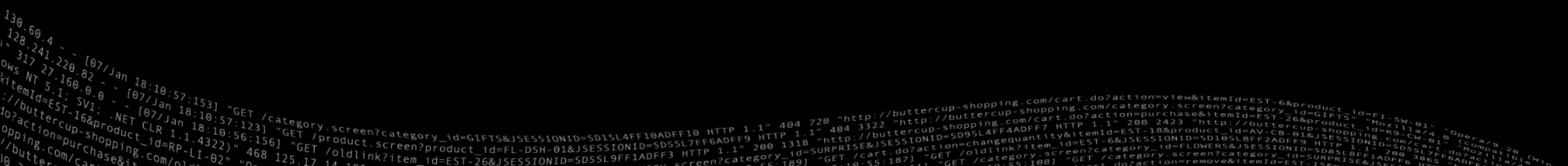
How'd We Do It?

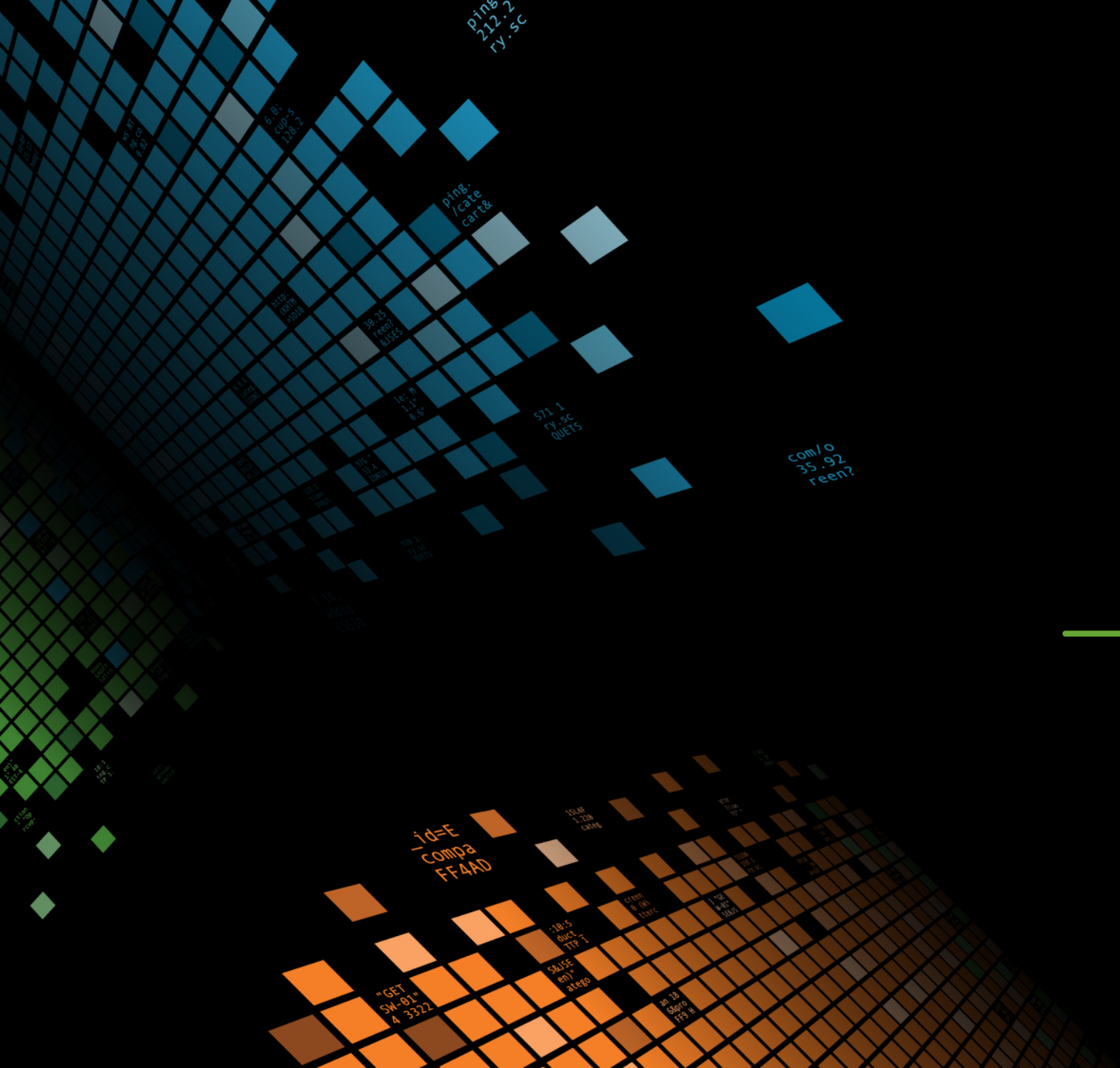
- We'll never tell

- Just kidding, that's why we're here!

Agenda

- About Us!
- Discuss the issues that caused Splunk to look for container based solutions
- Describe the tool goals for the problem set
- Give an overview of what the orchestration tool can do
- Give a brief demo of splunk setup and configuration
- Describe the architecture of the tool
- Describe the architecture of the containers
- Describe why we setup containers the way we did
- Question and Answer





About Us!

I Like To Pretend

(that I'm good at disc golf…)

- Developer at Splunk for 7 years
  - Splunk Apps -
    - ITSI (Up until 1.0)
    - VMWare (2.0-4.0)
    - NetApp (Guidance only)
    - ES (only on 2.1)
  - Infrastructure
    - ORCA
    - Eventgen

A Little About Brent

- Working in technology since 2001
- Splunker since 2013
  - Splunk App Development
    - ITSI, Splunk for VMWare, Splunk for NetApp
  - Infrastructure
    - ORCA
- I like rock climbing, half marathons, and automation

Splunk Is Hard?

Forward-Looking Statements

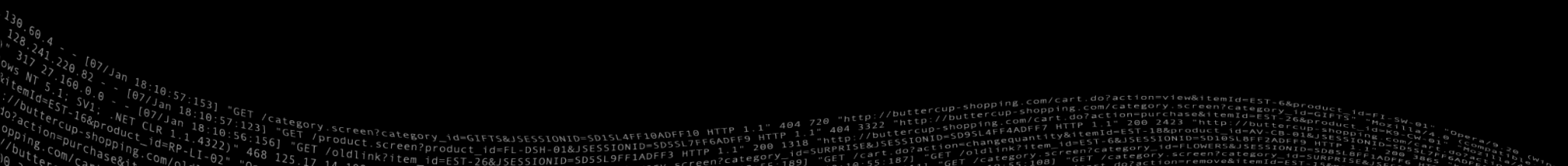
During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

splunk> .conf2017

With Great Power Comes Great Responsibility

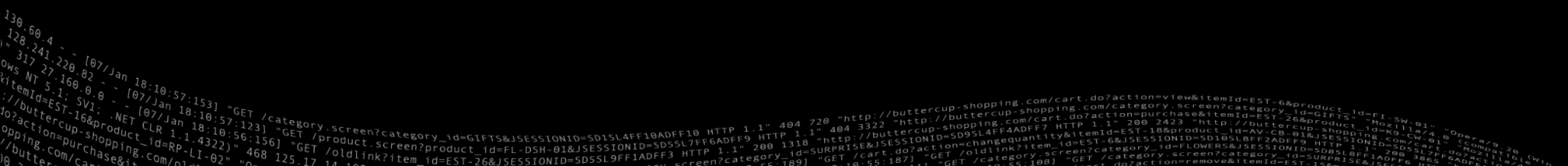
- Splunk's greatest strengths, are also it's biggest weaknesses
  - 100 ways to skin a cat (SPL)
  - Scaling Splunk from 1 rapid setup instance to clustering is challenging
  - Data ingestion done incorrectly forces reindexing
  - Writing apps for every Splunk architecture can prove challenging
    - (search head pooling, search head clustering, index clustering etc)
  - Takes several months to ramp up a new employee
  - Splunk is easy, Splunk at scale, isn't
  - Changing the architecture after it's built requires careful steps
  - Testing every Splunk Build in every configuration requires a massive infrastructure
  - Testing Splunk growing as data volume increases requires dynamic allocation of test nodes



Virtualization Styles

Platforms

- Tested Containers
  - kubernetes
  - docker UCP
  - docker swarm
  - (several others)
- Compared VM technologies
  - vmware
  - openstack
  - kvm

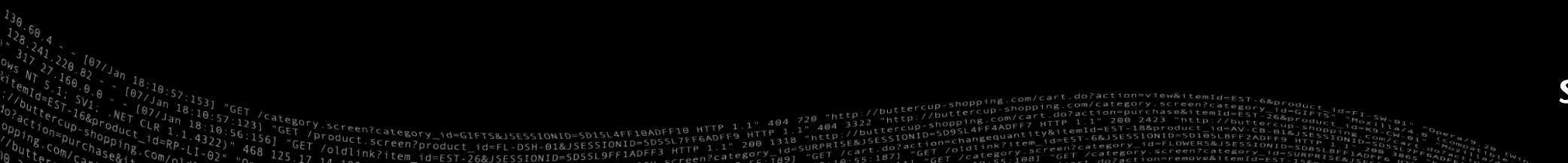


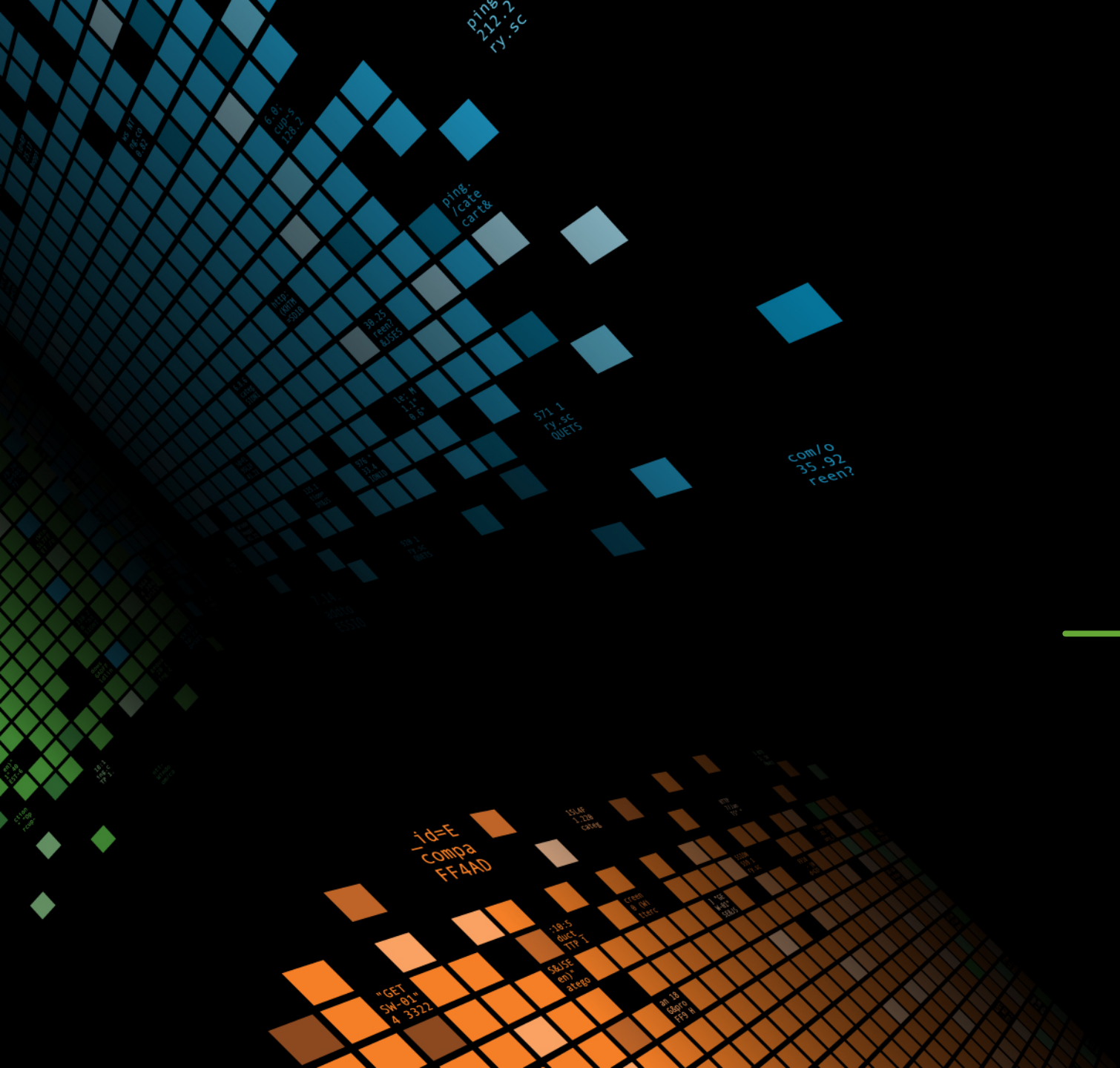
splunk> .conf2017

Everything Was Only 80% There…

- (Please note tools have features now, that they didn't back when we looked…)
- No single tool allowed complex orchestration of different image types
- No easy ways to isolate and extend
- No way to lock 1 container to 1 vm / host just for performance testing
- No speed degradation for the virtualization layer
- No way to run a standardized scripting language across windows and nix

- If one tool did one thing well, it lacked somewhere else…



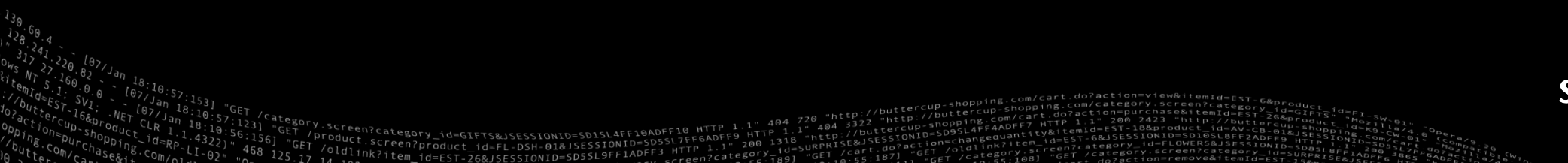


Enter ORCA

What Is ORCA?

Splunk Orchestration and Automation Layer

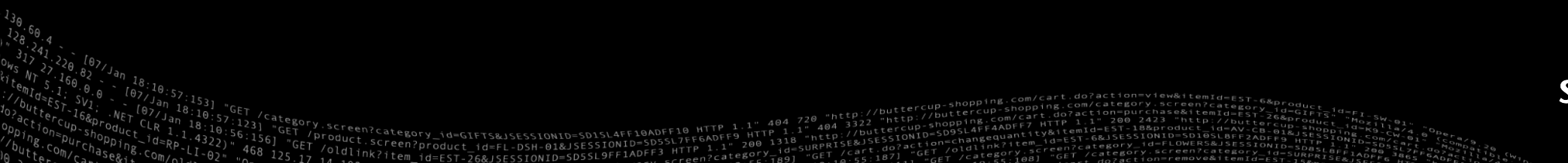
- ORCA enables users to quickly and correctly configure splunk test environments.
- ORCA can create different deployments very quickly
  - Less than one minute to bring up a standalone instance
  - About 3 minutes to bring up a generic search head cluster
  - Deployments can be local or deployed on UCP
- ORCA can create custom deployments very quickly and easily
  - Splunk cloud deployments
  - Deployments from feature branches for both apps and core builds
  - Additional customization can be provided by the user



splunk> .conf2017

Original Goals of ORCA

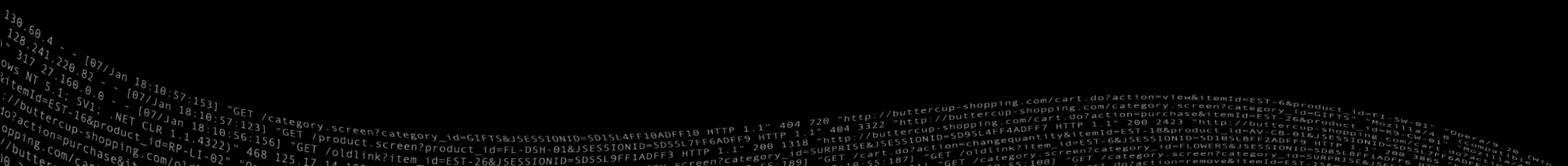
- Run our performance tests without needing to wait a week for a stack
  - These were usually 20 TB tests involving a large number of virtual machines
  - Often there were problems with configuration that introduced delays…
  - … because configuration was still being done mostly manually
- Lower transaction costs for developers to run tests in complicated environments
  - Developers would re-use VMs – and may have unclean environments
  - Configuration and setup problems would lead to longer testing times for features and bugfixes
- Automation!!



ORCA Design Principles

Ease of Maintenance, Ease of Use

- Strive for ease of use for early users
  - Documentation to get people off the ground quickly
- Strive for input flexibility for advanced users
  - Recipe style examples so that people know how to write advanced plays
  - Repository of sample plays to work with
- Strive for maintainability
  - Simple architectures, easy to read code. Anybody should be able to figure out what's going on
- No interactivity
  - Tools should be designed to be scripted and automated. Humans need not apply



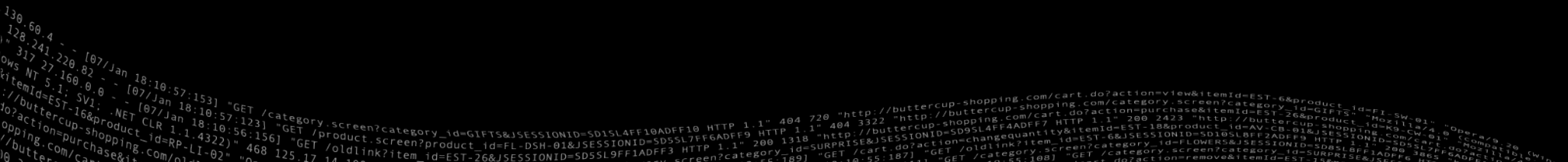
ORCA Concepts

Definitions and Workflow

splunk> .conf2017

Terminology/Definitions

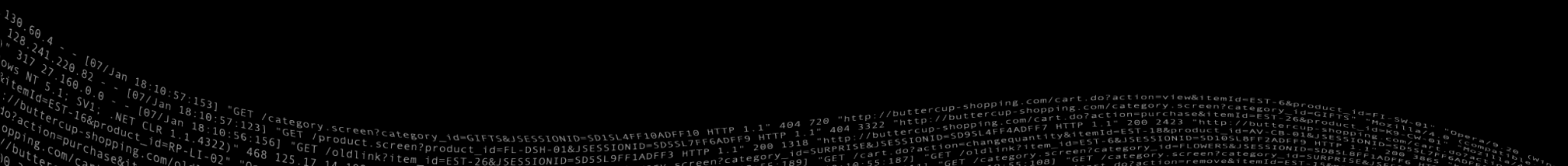
- Orchestrator container
  - This is the container that is run locally, and used to configure containers on the target
- Splunk ORCA container
  - These are the containers that run splunk, that are configured by the orchestrator container
- Key technologies utilized
  - Splunk
  - Docker
  - UCP
  - Openstack
  - Ansible



General Workflow And Scope

Cover the 85%

- We want to cover the largest set of general testing scenarios
  - Standalone instances
  - Search heads
  - Indexers
  - Apps
  - Heavy Weight Forwarders
  - … In any combination you want
- One stack per deployment
  - One search head cluster in the stack
  - For multi-site, and other advanced configurations, you can combine the networks



ORCA Features

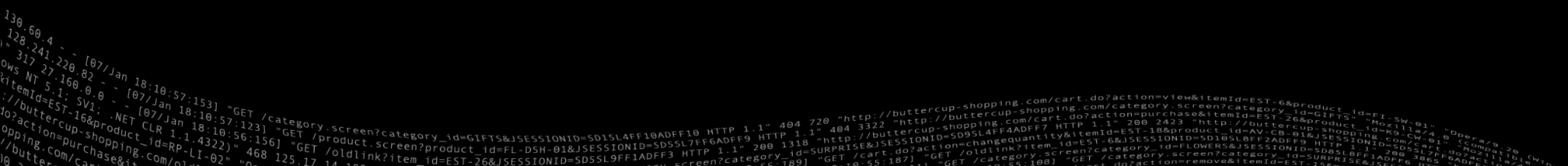
What ORCA can do for you

splunk> .conf2017

The Commands

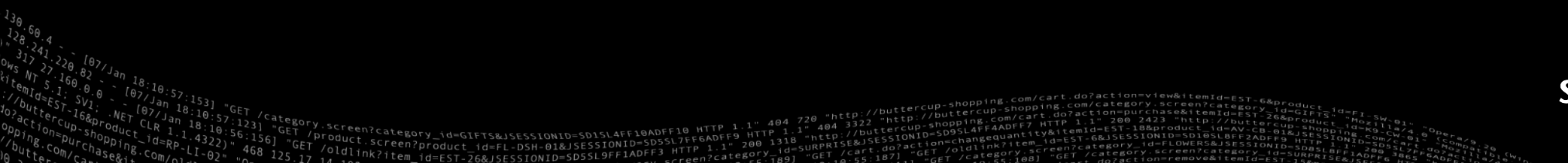
So far…

- Create Command
  - Create a stack
- Provision Command
  - Poorly named
  - Run ansible plays against the stack
- Destroy Command
  - Tear down a stack
- Show Command
  - Show deployment or container details
- Exec Command
  - Execute an action on the container
- Upgrade Command
  - Upgrades a stack
- Config Command
  - Configure the environment
- Start Command
  - Start the container or deployment (if stopped)
- Stop Command
  - Stop the container or deployment
- Build Command
  - Build a container image

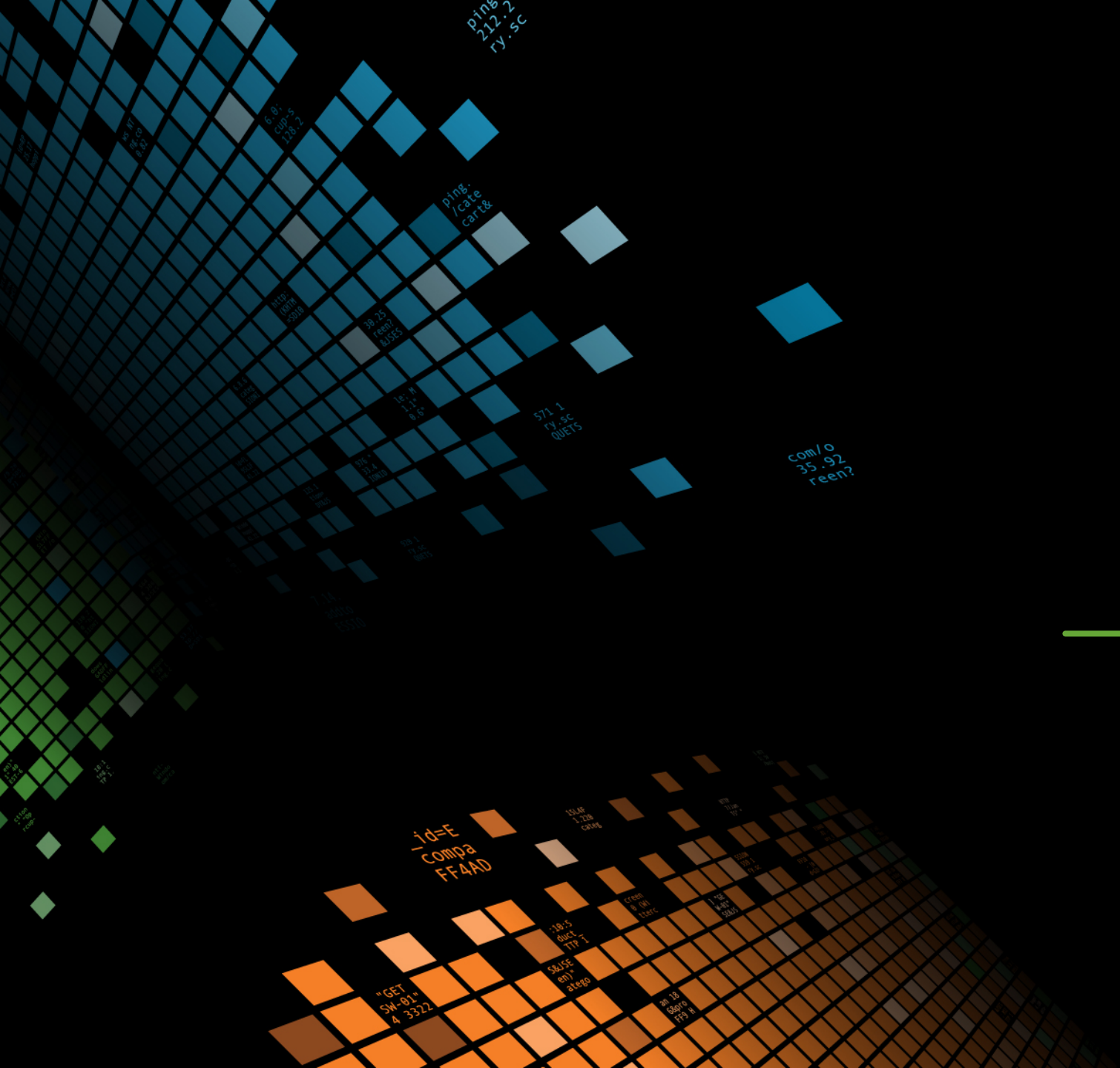


Custom Ansible Plays, Custom Services

- We can run any custom ansible play after provisioning using `--playbooks`
  - Instructions and Examples are provided on Confluence
  - We want to enable Mission Teams / Users to test a wide range of scenarios with automated, traceable configurations.
  - The provision command separates it out so that these steps can be run on demand
- We can create additional docker containers in the stack using `--services`
  - These containers are likely not splunk containers, and custom to the specific testing environment
- Combine these two, and you have a powerful provisioning and configuration tool.



splunk> .conf2017



Demo

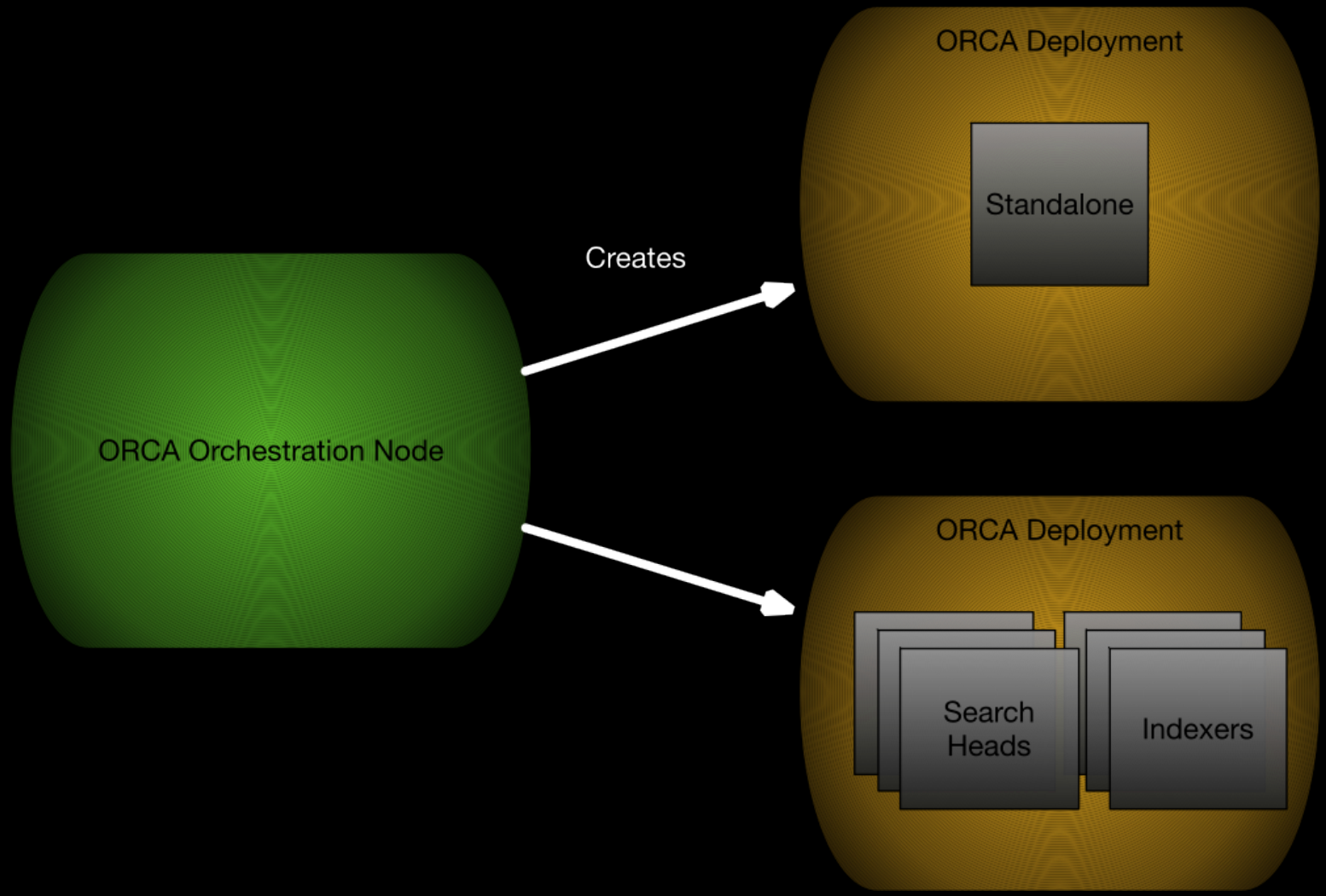
ORCA Architecture Overview

View from 30,000 feet

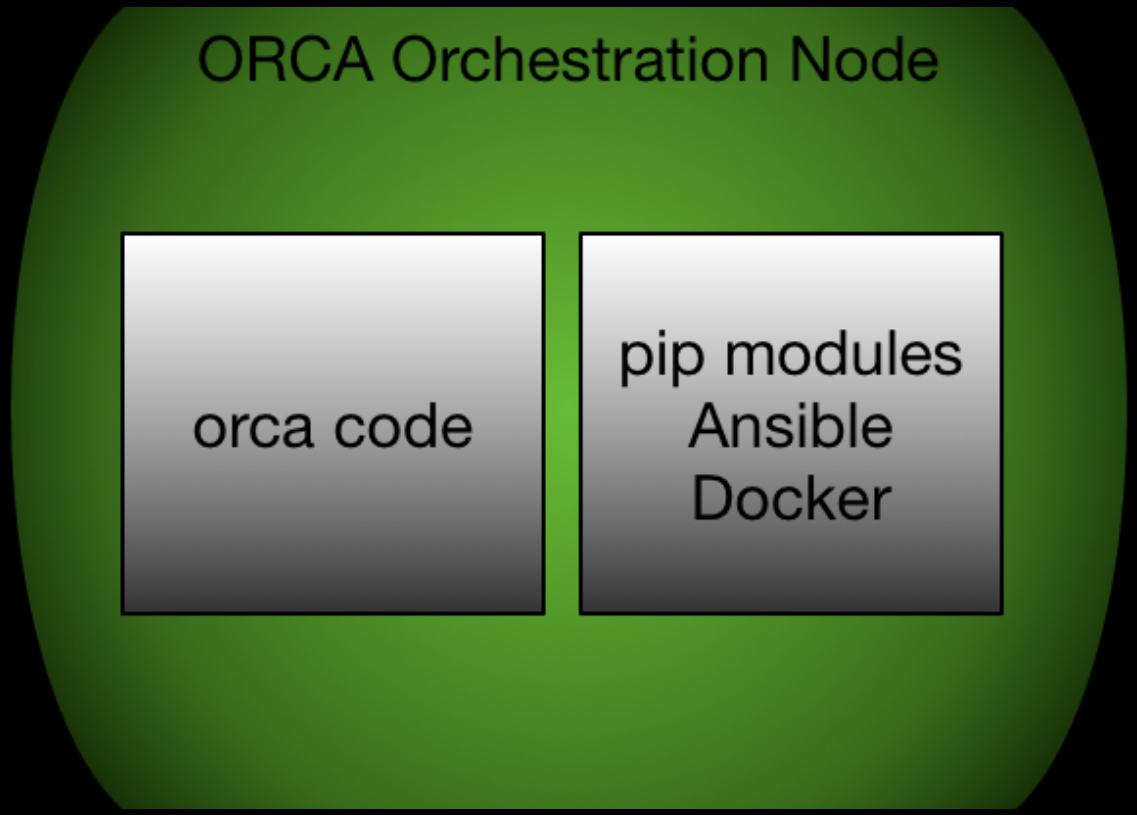
splunk> .conf2017

Introducing The Two Different Containers

The Orchestrator and the Deployment Containers



The Orchestrator Container

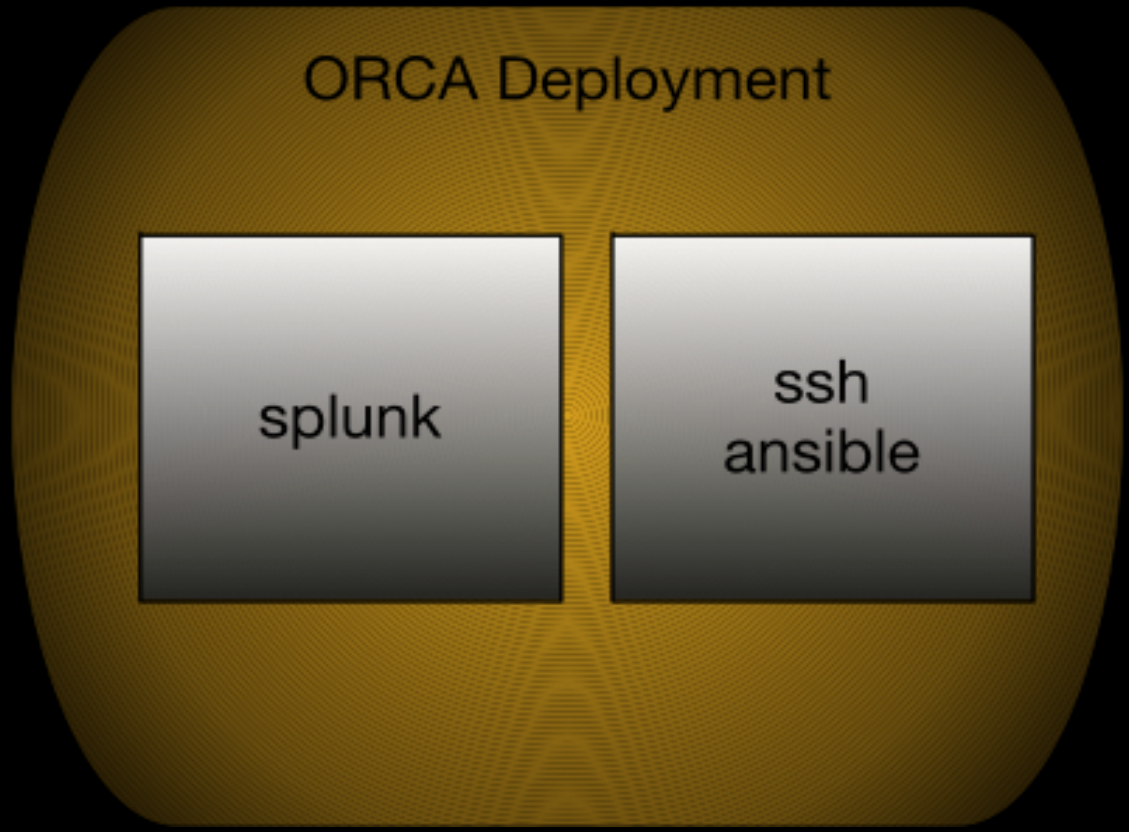


Internals

- The Orchestration Container
  - Executes nearly all of the code for deployment
- Why did we choose to ship as a container?
  - Keep installation dead simple
  - No dependency hell to walk through
  - Consistent environment

The Splunk Container

- The Splunk Container
  - Splunk
  - SSH – for ansible
    - Ansible can also be run from this node
    - Ansible itself
- Pretty heavyweight containers
  - We don't follow container best practices here, we're making them more like VMs



ORCA Splunk Container Ecosystem And Internals

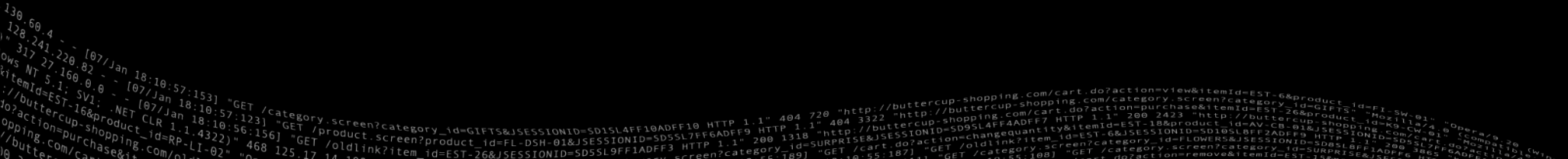
Stacking The Deck

splunk> .conf2017

Splunk Container Contents

What's installed

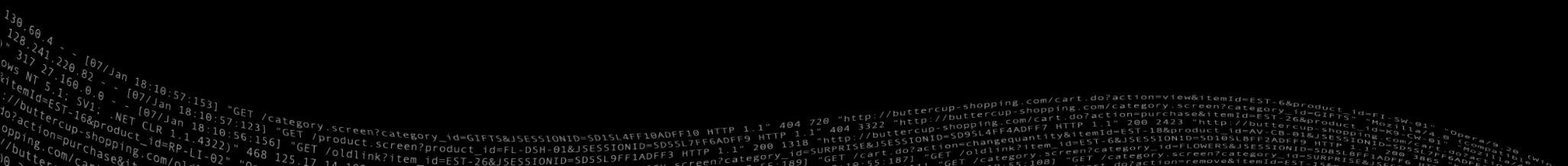
- /opt/splunk
  - Our favorite location for all of the splunk binaries, also $HOME
- sshd
  - Required for ansible
- python
  - Generally useful
- ansible
  - "Ansible mode" – run the plays from the Splunk Container instead of the Orchestrator. Good for working over a bad vpn connection.



Different Ways To Run The Splunk Container

How do we use our resources

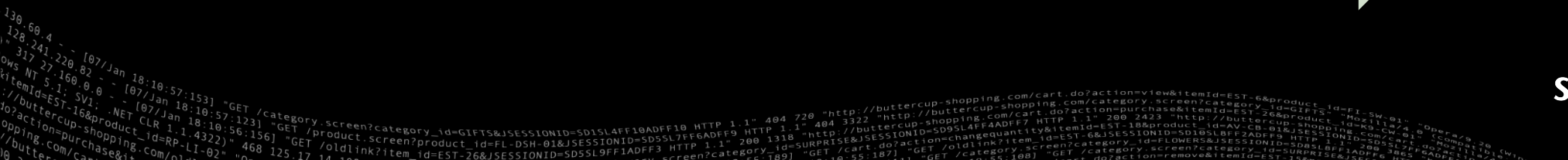
- UCP is our container environment
  - Currently 600 nodes
- Performance mode
  - One container per host
  - Allow the container to use all of the resources of the host
- Test mode
  - As many containers as we can binpack into the remaining nodes (in theory)
  - Different deployments may be on a single node or split between nodes
- Local mode (--local)
  - Not recommended for larger stacks
  - Avoids UCP altogether, good for bad network connections



Creating The Splunk Containers

General Workflow

- Provision compute resources. Usually has splunk binary
- Pre-ansible configuration. Copy SSH keys.
- Run ansible to complete splunk configuration.

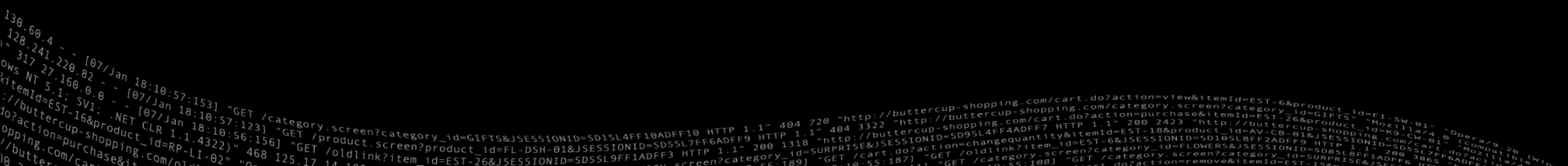


splunk> .conf2017

Why Did We Split It Up This Way?

Why not offload as much as we could into the static image

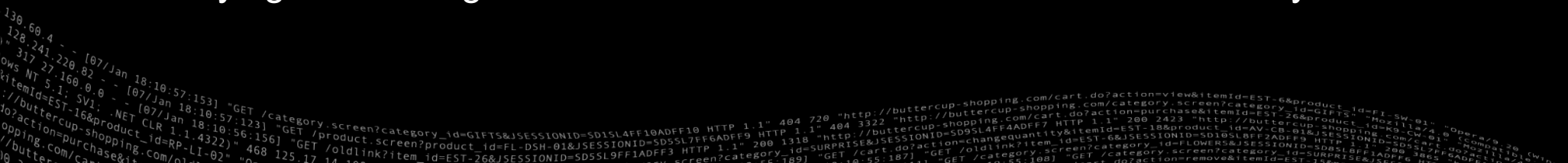
- FLEXIBILITY!
  - Testing environments can become chaotic very quickly, we want to anticipate any kinds of changes that a user wants
  - Sets up us nicely for customization
- Keep the container count minimal
  - Statically, it would be (Count of images)*(Number of Roles) = A lot
  - We don’t want to add build layers on top of every commit
- Splunk itself has some issues with configuration
  - Ansible is much better about retrying tasks that have a high rate of failure
- We do not want to rely too much on docker

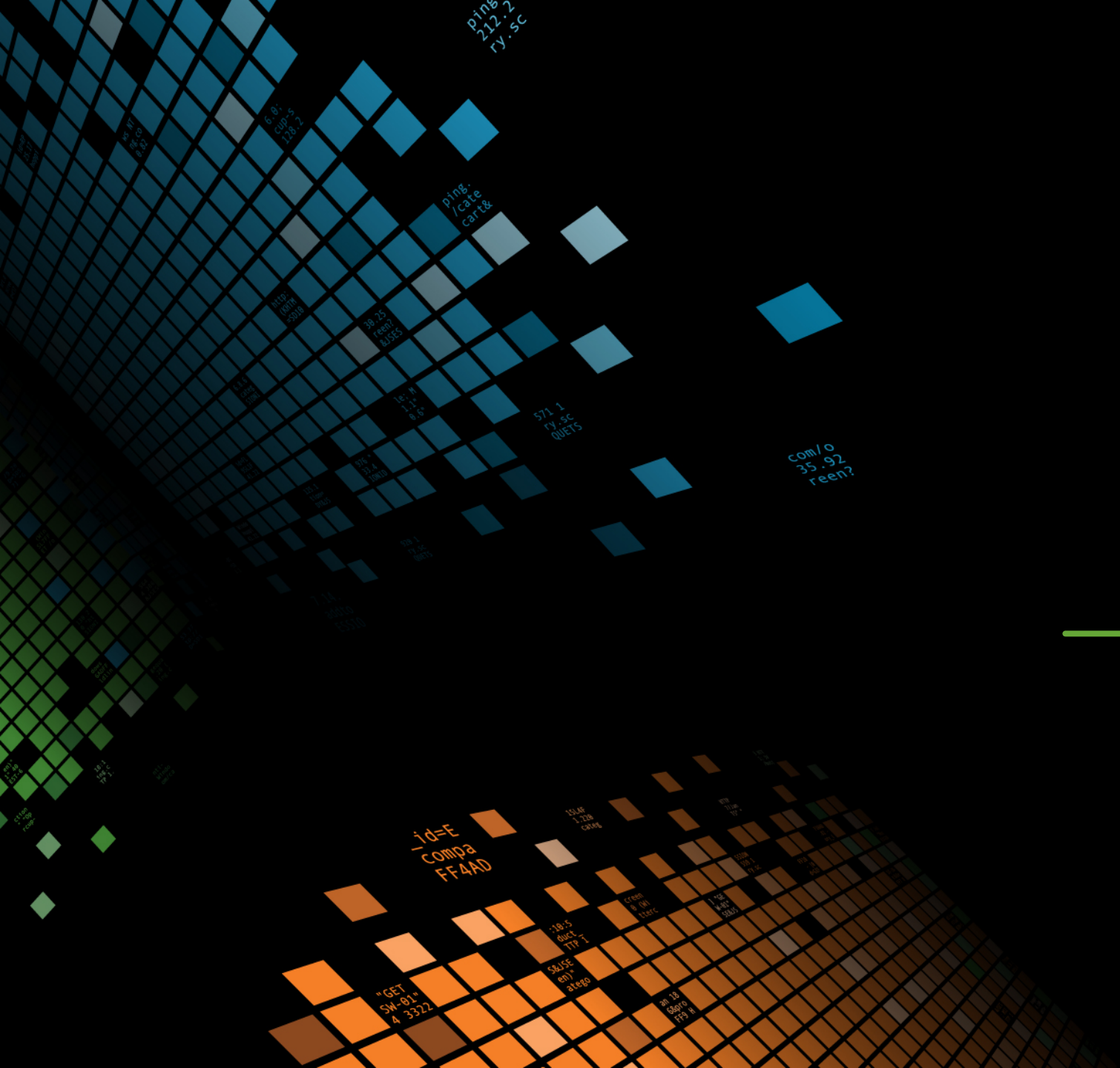


Executing The Ansible Code

Provision Compute Resources, then run Ansible to configure

- Ansible does the following for us
  - Configure the search head cluster
  - Configure individual nodes
- Install apps
- Configure apps
- Run custom plays against custom containers
  - Each type of custom container gets its own role
  - Easy to extend
- Each role determines what commands to execute on the host
  - A stack can consist of any number of roles
- This mostly follows ansible best practices
- The roles themselves are fairly resilient.
  - Retrying and waiting until resources become available when necessary



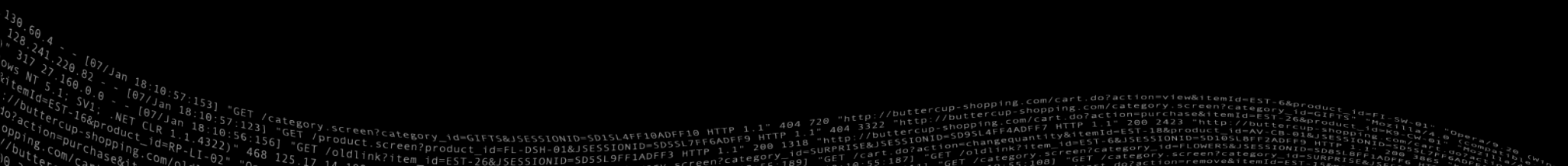


The Payoff

splunk> .conf2017

How'd We Do A 20 TB / Day Test?

- We ran 1 command:
  - orca create —sh 5 —idx 30 —eg 100
    - That command creates 135 containers on UCP
      - It does this by bringing up the local orchestrator
      - UCP schedules the containers on nodes
      - UCP then hands off the created containers back to the orchestrator
      - Ansible takes over and starts to provision the machines to their desired role
      - After the stack is provisioned, Ansible then tells the event-generators to start firing events

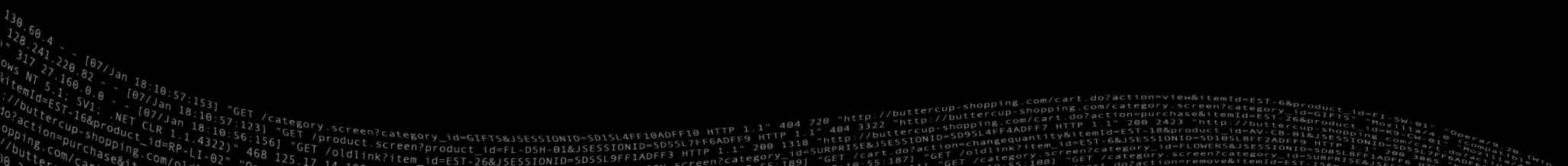


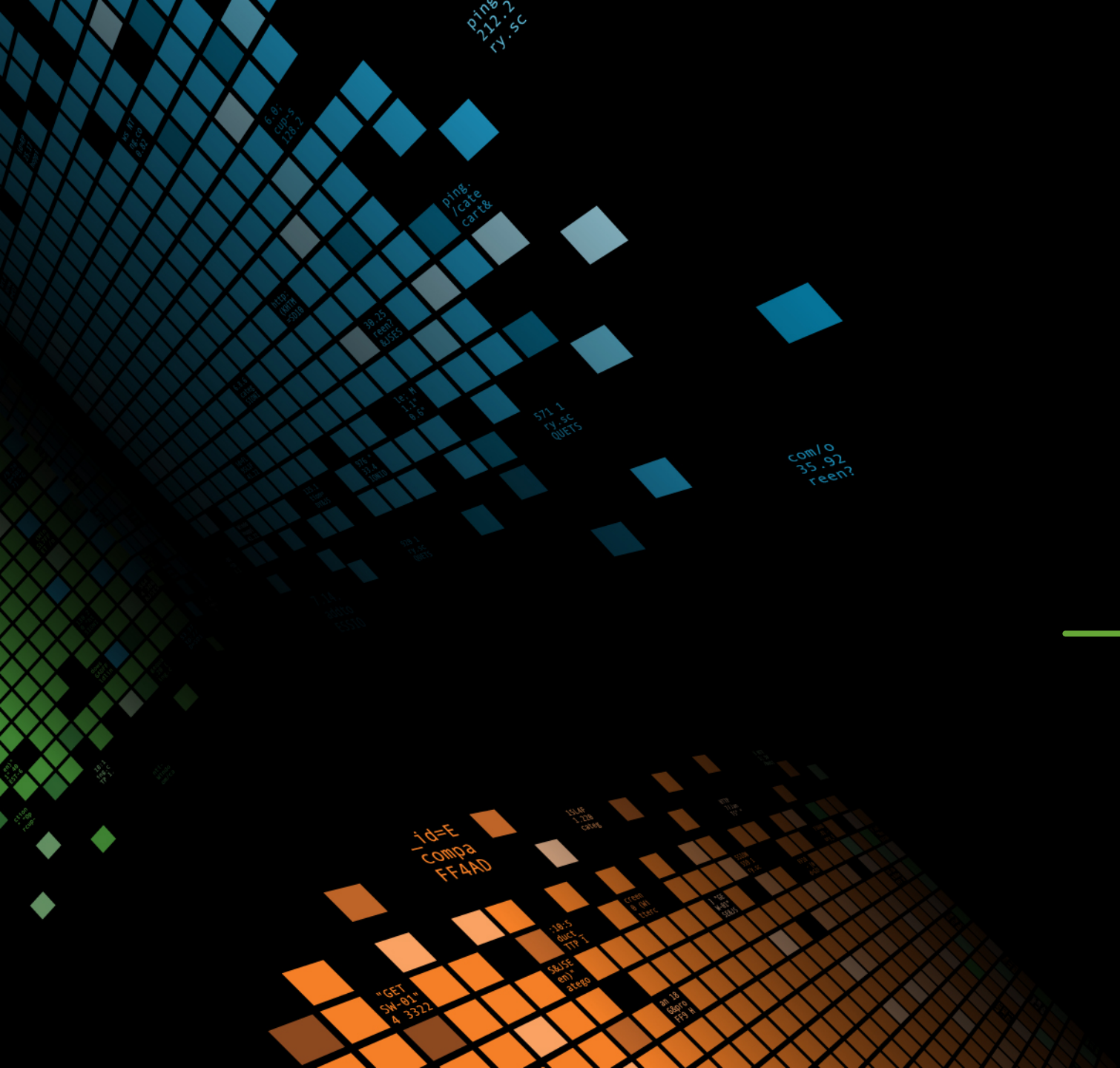
splunk> .conf2017

Results

- Produces a single method for recreation
- Repeatability
- No Prior Splunk knowledge needed for stack creation
- Full cluster setup with load balancing in front of indexers, in under 15min
- Testing at larger scales on all functional testing allows us to catch bugs before you hit them

- End User's Reaction: "I wish I had a tool like this 2 years ago"





Questions?

splunk> .conf2017

Thank You

Don't forget to rate this session in the .conf2017 mobile app

splunk> .conf2017