Splunk Multi-Deployment Server Architecture

Joe Cramasta & Kate Lawrence
Senior Engineer & Platform Architect
Splunk

splunk> .conf19

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.

What are we talking about today?

.conf19
splunk>

Kate

- Almost 20 years experience in infrastructure management, systems operations, security & big data architecture.
- Spent 6 years with Comcast
  - Principal Engineer (Splunk)
  - Senior Manager of Engineering & Software Development
    - Focus on open source integrations with existing data platforms
- Inaugural SplunkTrust member & 2013 Revolution Award Winner (Innovation)
- Joined Splunk ~18 months ago as Platform Architect in the Global Engineering team.

Joe

- 8 years at Comcast
  - Principal Engineer (Splunk)
- < 2 years at Splunk
  - Premium Cloud Support Engineer
  - Pre-Sales Engineer
- 2014 & 2015 Presenter at Splunk .conf
- 2015 Splunk Ninja Revolution Award Winner

What is this presentation going to cover?

- Covered
  - Simplified model for deploying & operating multiple Splunk Deployment Server nodes
  - A Git based method for using version control with multiple Splunk Deployment Servers
  - High-level usage of Ansible playbooks to update multiple Splunk Deployment Servers with changes from Git
  - Trade-offs between an multiple Deployment server method vs single-host systems
- Not Covered
  - Intro to Git
  - Intro to Ansible
  - Tiered Deployment servers (no longer supported after 7.0+)
  - Forwarder Management UI

Deployment Server at Scale

Topic breakdown

- Introductions
- Deployment Server Overview & Single node limitations
- Implementing a Multi-DS strategy
  - Architecture Overview
  - DNS vs Load Balancers
  - Using Git & Ansible for changes & updates
- Reporting & Monitoring
  - Peering & Monitoring Console
  - Reports & APIs
- Conclusion & QA

What is Deployment Server?

You use the Splunk **deployment server** to distribute content and configurations (collectively called **deployment apps**) to **deployment clients**, grouped into **server classes**.

Deployment apps can be full-fledged **apps**, such as those available on Splunkbase, or they can be just simple groups of configurations.

What is Deployment Server?

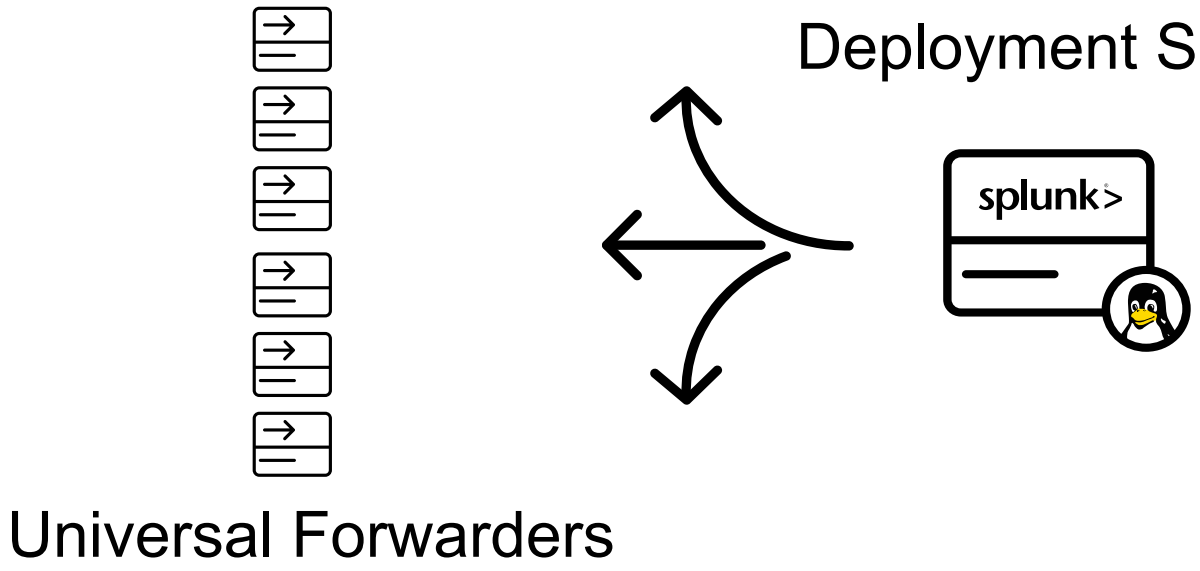
continued…

Splunk's deployment server is Publisher/Subscriber application that uses an (internal) Broker to negotiate sessions with Universal Forwarders.

The Universal Forwarder will PhoneHome to a Deployment Server node & validate the latest app/configuration against the current checksum value to determine if there are changes

Single Deployment Server

Topology

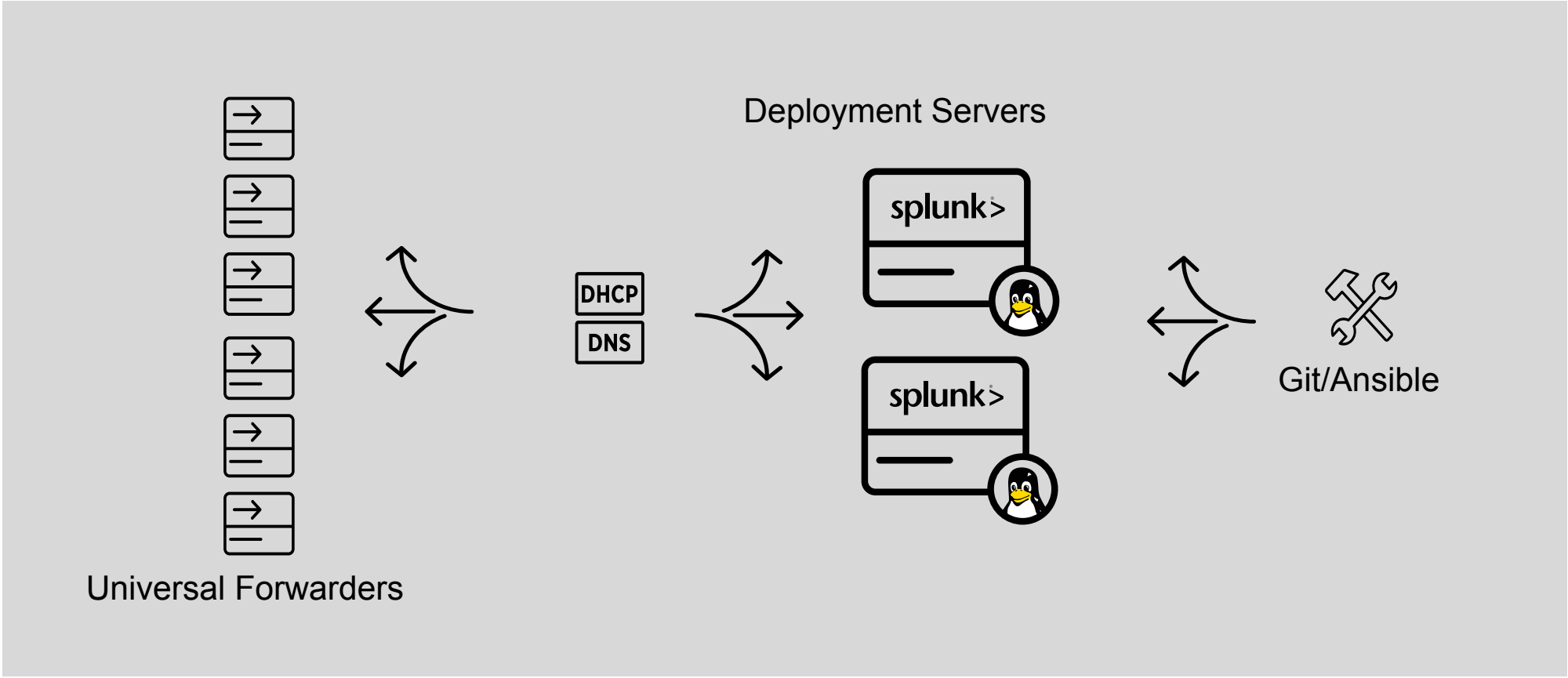


Limits:

- Single Threaded
- # of Inbound TCP Connections
- Capacity:
  - Linux DS – **5000 UFs**
  - Windows DS – **2500 UFs**

Multi-Deployment Server (MDS)

Topology



Multi-Deployment Server (MDS)

Topology

- Universal Forwarders have a **targetUri** (deploymentclient.conf) that is set to to a **DNS RR A record**

```
[deployment-client]
[target-broker:deploymentServer]
targetUri=deploy.klgsplunk.com
```

- Deployment Server nodes are configured with
  - **crossServerChecksum=true**
  - This is disabled by default
  - In this configuration UI components should not be used.

- Git/Ansible is used on the backend to store the deployment server content
  - This requires familiarity with version control & automation frameworks
  - These components should also feed output/logs into Splunk for monitoring

Multi-Deployment Server Architecture

DNS vs LB Configurations

- Splunk's recommendation is to use a DNS Round Robin A record to rotate between multiple DS nodes.
  - This approach has been shown to work more consistently than using a Load Balancer/VIP
  - Load Balancers *can* certainly be used but additional network review is typically required since:
    - DS nodes need to be able to negotiate sessions with the client at layer 3
    - NAT/Firewall configurations can cause Universal Forwarders to fail phoneHome actions
    - The Universal Forwarder (currently) doesn't send the appropriate "x-FWD" headers for HTTP proxies to work consistently

Multi-Deployment Server Architecture

DNS vs LB Configurations

```
$TTL 86400
@   IN  SOA     ns1.klgsplunk.com. root.klgsplunk.com. (
        2013042201  ;Serial
        3600        ;Refresh
        1800        ;Retry
        604800      ;Expire
        86400       ;Minimum TTL
)
; Specify our two nameservers
                IN      NS              ns1.klgsplunk.com.
; Resolve nameserver hostnames to IP, replace with your two droplet IP addresses.
ns1             IN      A               10.0.0.19

; Define hostname -> IP pairs which you wish to resolve
sds1    86400    IN  A 10.0.0.82
sds2    86400    IN  A 10.0.0.156
sds3    86400    IN  A 10.0.0.127
deploy  3600     IN  A 10.0.0.156
                 IN  A 10.0.0.127
```

Figure 1 - local named zone configuration

```
[deployment-client]
clientName=dc
phoneHomeIntervalInSecs=3600
[target-broker:deploymentServer]
targetUri=deploy.klgsplunk.com:8089
```

Figure 2 - local deploymentclient.conf configuration

- Example:
  - Universal Forwarders are configured to use a DNS – Round Robin A record deploy.klgsplunk.com
  - This A record will rotate the host record on an 1 hour (3600 second) schedule between hosts sds2 & sds3
  - This should match or be less than the polling interval set at the UFs
    - 1 hour or longer works well consistently in testing

Multi-Deployment Server

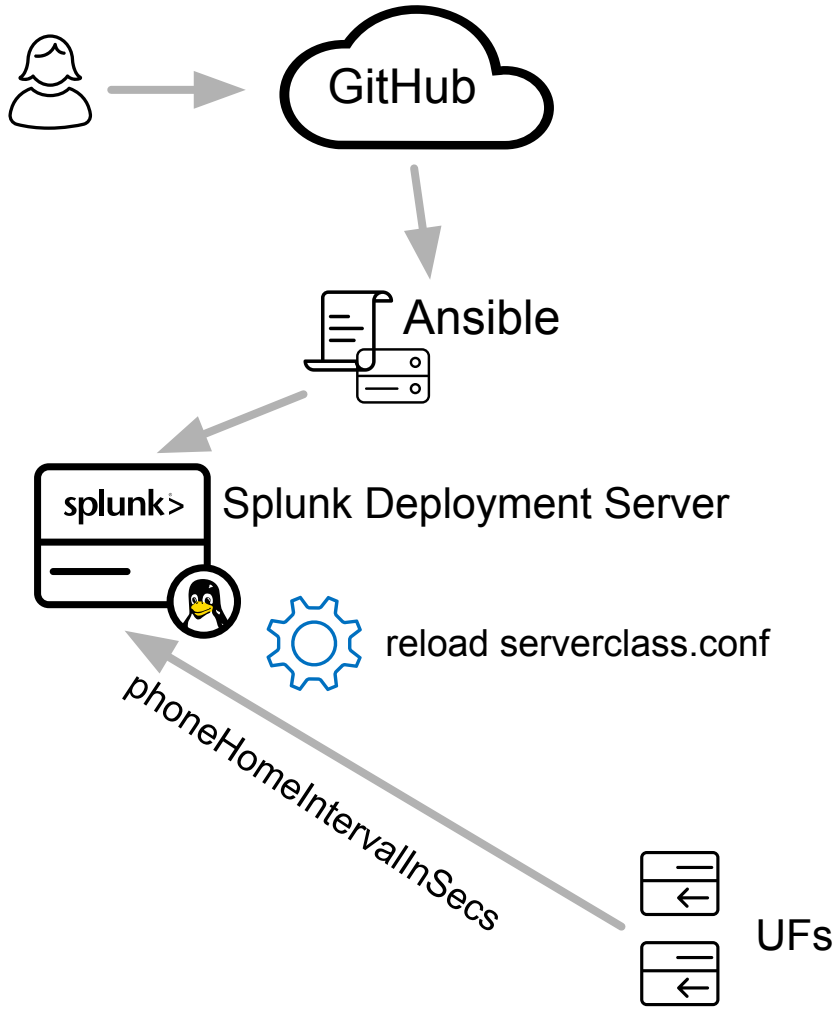
Considerations

- Pros
  - MDS nodes can be scaled up to support 10s of thousands of hosts
  - Splunk UF & Deployment Server node reporting available via logs & REST API
  - Introduction of Git repository allows for
    - Version control of deployment-apps & configurations
    - Rollback features
    - Peer Review & Change Control
- Cons
  - Managing multiple DS nodes requires the use of tooling outside of Splunk:
    - Integration of Git repository requires additional checks on configurations prior to deployment
    - Ansible playbooks & SSH keys required
  - The DS nodes are still single-threaded
  - Forwarder Management UI should not be used in this model.
  - Managing the serverclass.conf directly through the file
  - Requires additional checks & balances

Multi-Deployment Server

Sample Workflow

- All changes are checked into Git & merged to a golden or master branch
  - Ansible playbook (on a per DS node basis)
    - pulls configurations from the Git repository
    - update all DS nodes with the same configuration
    - reload the deployment server
  - UFs check-in based on phoneHome intervals
    - UFs are provided updates based on the serverclass.conf configuration



MDS Monitoring

Peering & Monitoring

- Use A Monitoring Console Server

  - All Splunk Deployment Server nodes should be peered & designated as deployment-servers
  - All Splunk Deployment Servers nodes should have a custom group name assigned to them, for example: **mds**
    - REST command searches can be targeted to all MDS nodes (splunk_server_group)

	Instance (host)	Instance (serverName)	Machine	Server roles	Custom groups	Indexer Cluster(s)	Search Head Cluster(s)	Monitoring	State
>	ip-10-0-0-127.ec2.internal	ip-10-0-0-127.ec2.internal	ip-10-0-0-127.ec2.inte...	Deployment Server	mds			✓ Enabled	Configured
>	ip-10-0-0-156.ec2.internal	ip-10-0-0-156.ec2.internal	ip-10-0-0-156.ec2.int...	Deployment Server	mds			✓ Enabled	Configured

What Should We Monitor For?

CHECKSUMS!

- All serverclass.conf use **crossServerChecksum=true**
- All deployment server **apps have the same checksums**
- All deployment servers have the same **serverclass.conf**

REST ENDPOINTS FTW!

Monitor serverclass.conf uses crossServerChecksum = true

/services/configs/conf-serverclass

Example REST command search from Monitoring Console:

| rest splunk_server_group="dmc_customgroup_mds" /services/configs/conf-serverclass

crossServerChecksum enabled on both deployment servers

deployment_server	crossServerChecksum
ip-10-0-0-127.ec2.internal	Enabled
ip-10-0-0-156.ec2.internal	Enabled

REST ENDPOINTS FTW!

Monitor all deployment server apps have the same checksums

/services/deployment/server/applications

Example REST command search from Monitoring Console:

| rest splunk_server_group="dmc_customgroup_mds" /services/deployment/server/applications

Matching app checksums across deployment servers

Deployment Server Application Checksums

title	deployment_servers	checksum	status
fwd-app	ip-10-0-0-127.ec2.internal ip-10-0-0-156.ec2.internal	8830064211521554482	Matching
fwd-app-2	ip-10-0-0-127.ec2.internal ip-10-0-0-156.ec2.internal	7713279339441192731	Matching

What About serverclass.conf?

Monitor all deployment servers have the same serverclass.conf

- Checksums are not available from any endpoint for serverclass.conf
- Could use the /services/configs/conf-serverclass endpoint and do some SPL foo to do a stanza by stanza setting comparison

What About serverclass.conf?

Monitor all deployment servers have the same serverclass.conf

- **Put serverclass.conf in its own app and let Splunk take care of creating a checksum for it, just like any other app.**

- **Leverage the same endpoint that we used for**

Example REST command search from Monitoring Console:

| rest splunk_server_group="dmc_customgroup_mds" /services/deployment/server/applications

Severclass Checksum

title	deployment_servers	checksum	status
serverclass	ip-10-0-0-127.ec2.internal ip-10-0-0-156.ec2.internal	14628466518199378430	Matching

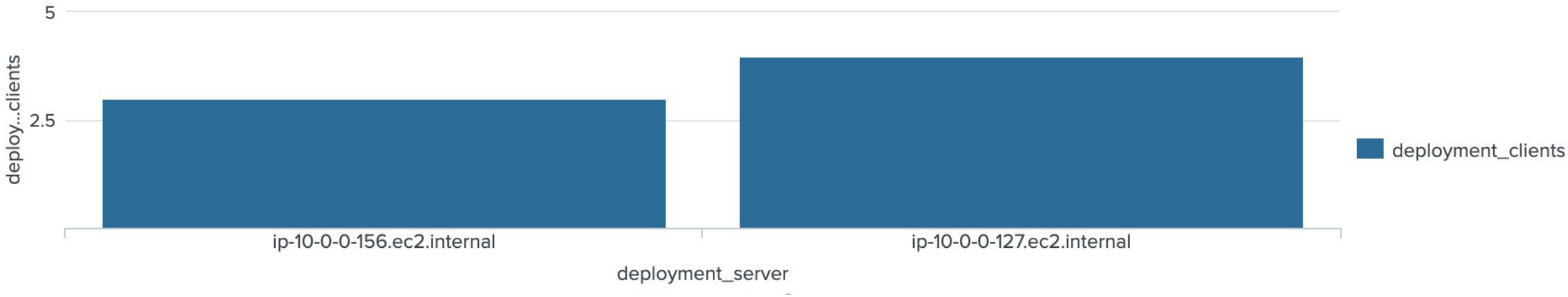
What Should We Monitor For?

Deployment Client Distribution

/services/deployment/server/clients

Example REST command search from Monitoring Console:

| rest splunk_server_group="dmc_customgroup_mds" /services/deployment/server/clients



Reference Links

- https://github.com/klawrencegupta-splunk/splunk_mds
- https://docs.splunk.com/Documentation/Splunk/7.3.2/Updating/Deploymentserverarchitecture
- https://docs.splunk.com/Documentation/Splunk/7.3.2/Updating/Deploymentserverarchitecture#Summary_of_key_terminology
- https://docs.splunk.com/Documentation/Splunk/7.3.2/RESTREF/RESTdeploy

splunk>

Thank You!

Go to the .conf19 mobile app to

RATE THIS SESSION