

EXTENDING AUTOMATION TOWARD SELF-SERVICE CDNS

Sept, 29th, 2020



BIO

JONATHAN GRAY

Comcast Content Delivery Network Team
Site Reliability Engineer 4

Twitter : @jhg03a

GitHub : @jhg03a

traffic-control-cdn.slack.com : @jhg03a

jhg03a@apache.org

https://about.twitter.com/en_us/company/brand-resources.html
<https://github.com/logos>
<https://slack.com/media-kit>
https://commons.wikimedia.org/wiki/File:Antu_mail-folder-sent.svg



PATHS FORWARD

CDN-in-a-Box (CIAB)

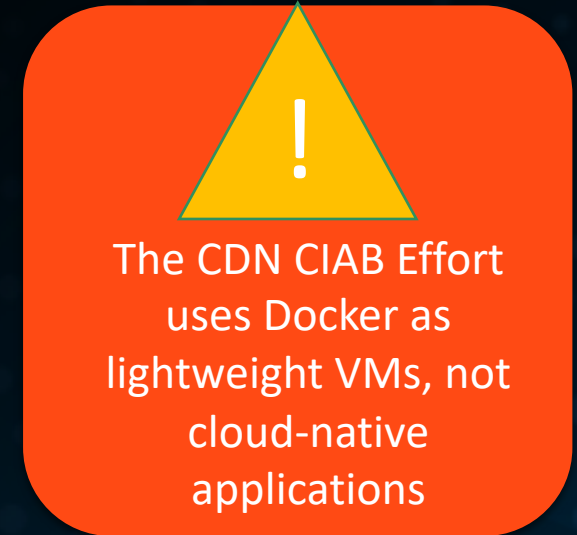
- Developers who need a local full-stack to aid in development
- Common infrastructure to enhance our GitHub Actions CI
- New users without prior CDN experience
- Functional Demonstration

https://traffic-control-cdn.readthedocs.io/en/latest/admin/quick_howto/ciab.html

Ansible-based Lab Deployments

- DevOps engineers building more production-like lab environments
- New Greenfield production deployments
- Developers needing a test environment that cannot be modeled on single host resources

Partial collaboration with our Open Source community on commonalities between participants



TLDR RECAP FROM APACHECON 2019

SILDES: <https://tinyurl.com/AutomatingATCSlides>

VIDEO: <https://tinyurl.com/AutomatingATCVideo>

ENVIRONMENT ABSTRACTION LAYERS

NOT CDN-OUT-OF-THE-BOX

Every abstraction layer comes at a price; some are more expensive than others. Lower costs through reuse of existing tools/skillsets.

| | Responsibilities | Example Technologies | |
|-----------------------|--|---|-------------------|
| Application Layer | <ul style="list-style-type: none"> • ATC Components • Application Monitoring • Data Visualization | <ul style="list-style-type: none"> • Ansible push • Shell script | Execution Wrapper |
| Steady-state OS Layer | <ul style="list-style-type: none"> • OS Users/Groups • Package Repositories • Host-based Firewalls • Kernel Optimization | <ul style="list-style-type: none"> • Puppet • Chef • Salt • Ansible Tower • Ansible-pull | |
| Provisioning Layer | <ul style="list-style-type: none"> • DNS • Network • Compute • RAID | <ul style="list-style-type: none"> • Terraform • VinylDNS • Foreman • MaaS | |

PROVISIONING

HOST-SPECIFIC ISO VIA TRAFFIC PORTAL

PRO

- No extra network dependencies
- Server identity baked into ISO
- Can pull in form data from TO Server List

CON

- Tedious & error prone at scale
- Requires all TO hosts to contain same kickstart files

`generate` is an optional executable to which TrafficOps will delegate the ISO creation process to after creating `ks_scripts/* .cfg`.

Multiple kickstart templates & OS versions are supported via `osversions.json`

EXAMPLE KICKSTART LAYOUT

```
./centos-kickstart
├── generate
├── isolinux
│   └── ...
├── ks
│   └── ks.cfg
├── ks_scripts
│   ├── ...
│   ├── disk.cfg
│   ├── mgmt_network.cfg
│   ├── network.cfg
│   ├── password.cfg
│   └── state.out
```

UNIVERSAL ISO WITH TC_NETCONFIG

PRO

- One ISO for all hosts, reducing error and improving deployment pace
- Continuous network identity maintenance via TrafficOps
- ISO Creation process is separate from TrafficOps

CON

- Requires IPv6 Autoconf RA

RESOURCES

- GitHub: <https://github.com/Comcast/tc-netconfig>
- ApacheCon 2019 Presentation: <https://tinyurl.com/tcnetconfig-video>
- ApacheCon 2019 Slides: <https://tinyurl.com/tcnetconfig-slides>

UNIVERSAL ISO WITH LAB MANAGER

PRO

- Leverages majority of work needed for tc_netconfig
- Solves Chicken/Egg problem between tc_netconfig & TrafficOps
- ISO Creation process is separate from TrafficOps

CON

- Requires IPv6 Autoconf RA
- Requires custom coding to create valid ifcfg-* & ifroute* files

This option is for non-production uses only



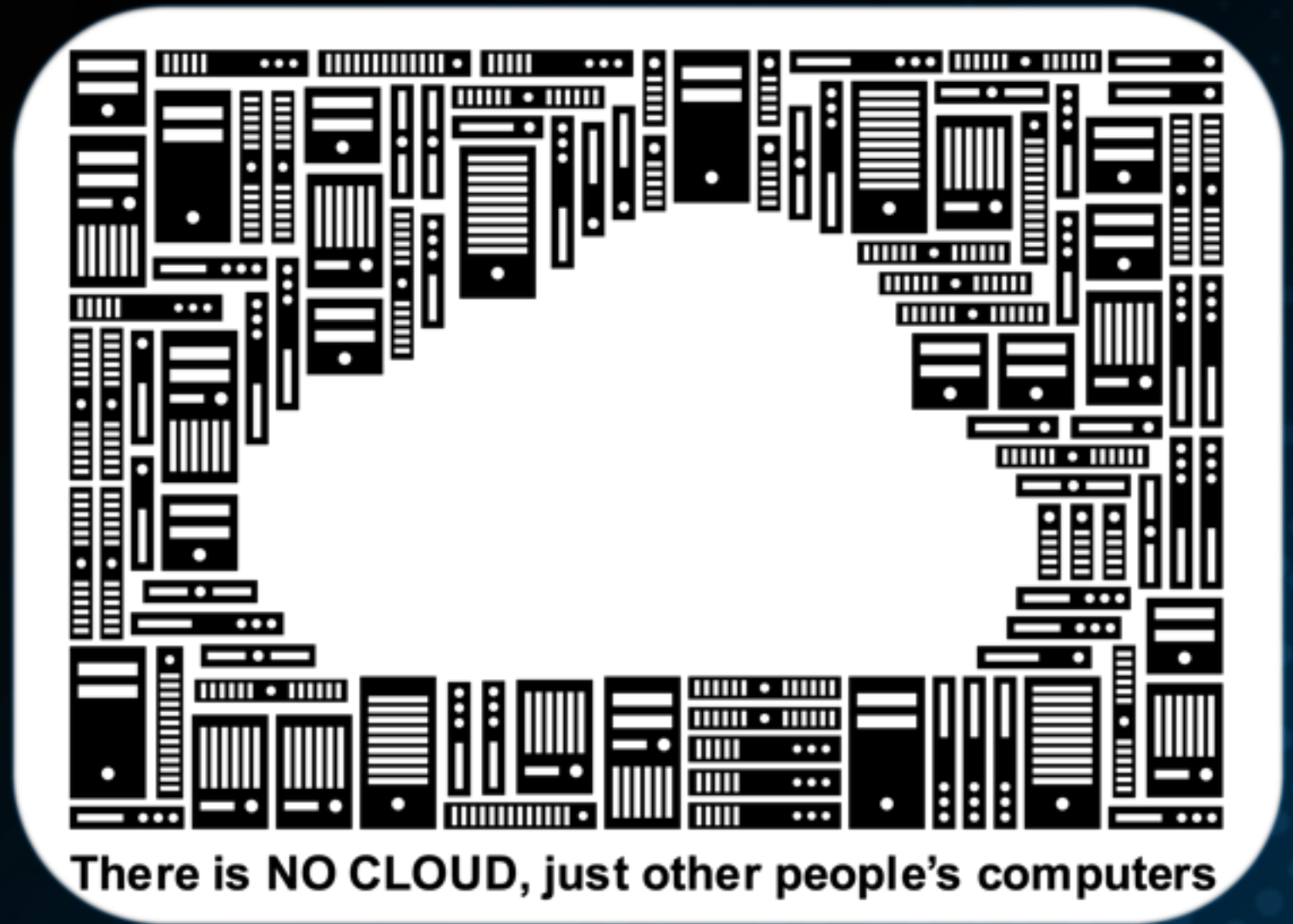



PHYSICAL DEPLOYMENT

CLOUD

TOOLING

- [HashiCorp Terraform](#)
- [VinyIDNS](#)
- [OpenStack](#)
- [Cloud-Init](#)

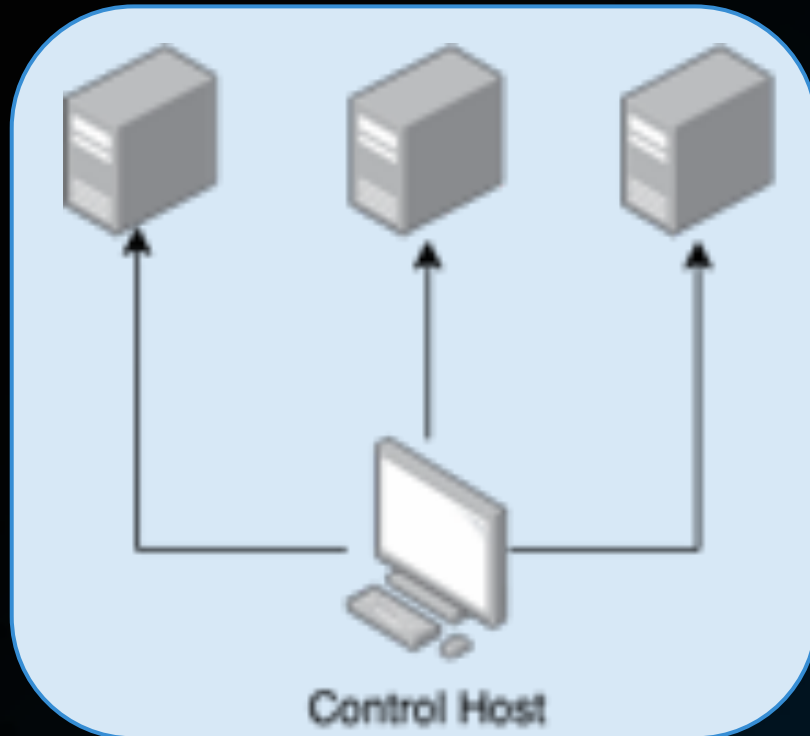




STEADY STATE

ANSIBLE WORKFLOWS

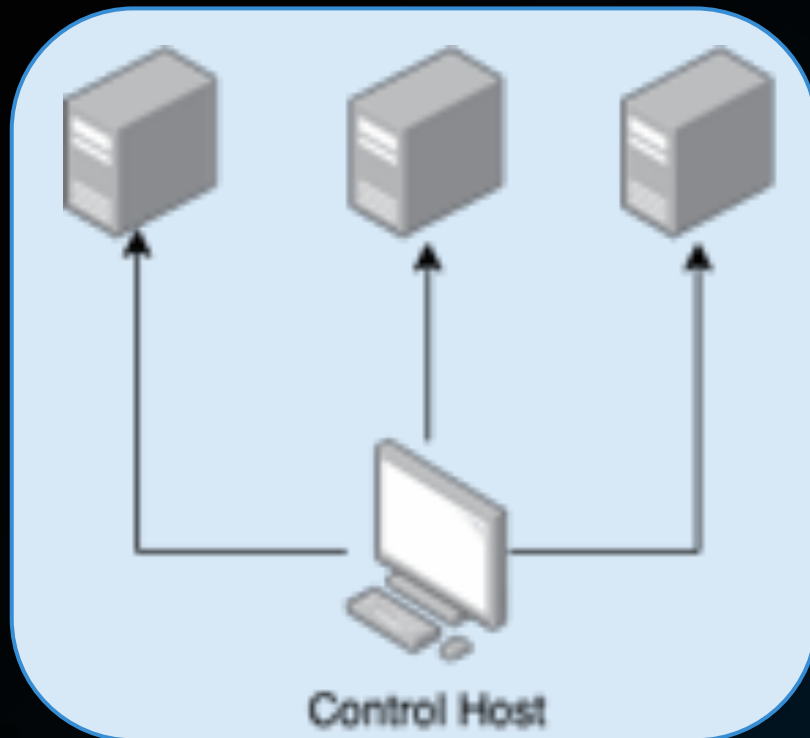
ANSIBLE (PUSH)



“Do this”

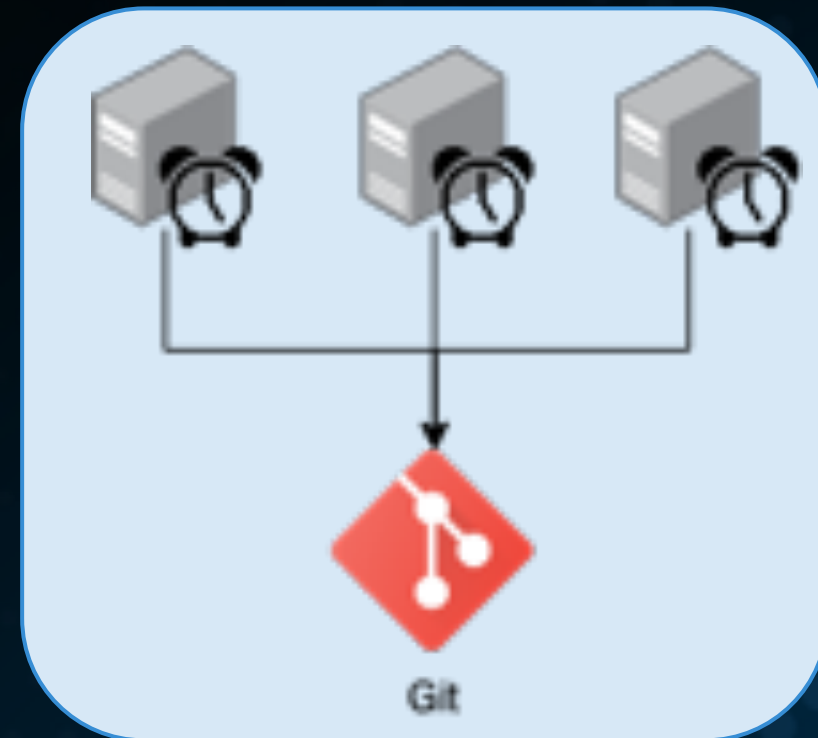
ANSIBLE WORKFLOWS

ANSIBLE (PUSH)



“Do this”

ANSIBLE-PULL



“Do what applies”

TLDR RECAP FROM APACHECON 2019

SLIDES: <https://tinyurl.com/AutomatingATCSlides>

VIDEO: <https://tinyurl.com/AutomatingATCVideo>

ENVIRONMENT ABSTRACTION LAYERS

NOT CDN-OUT-OF-THE-BOX

Every abstraction layer comes at a price; some are more expensive than others. Lower costs through reuse of existing tools/skillsets.

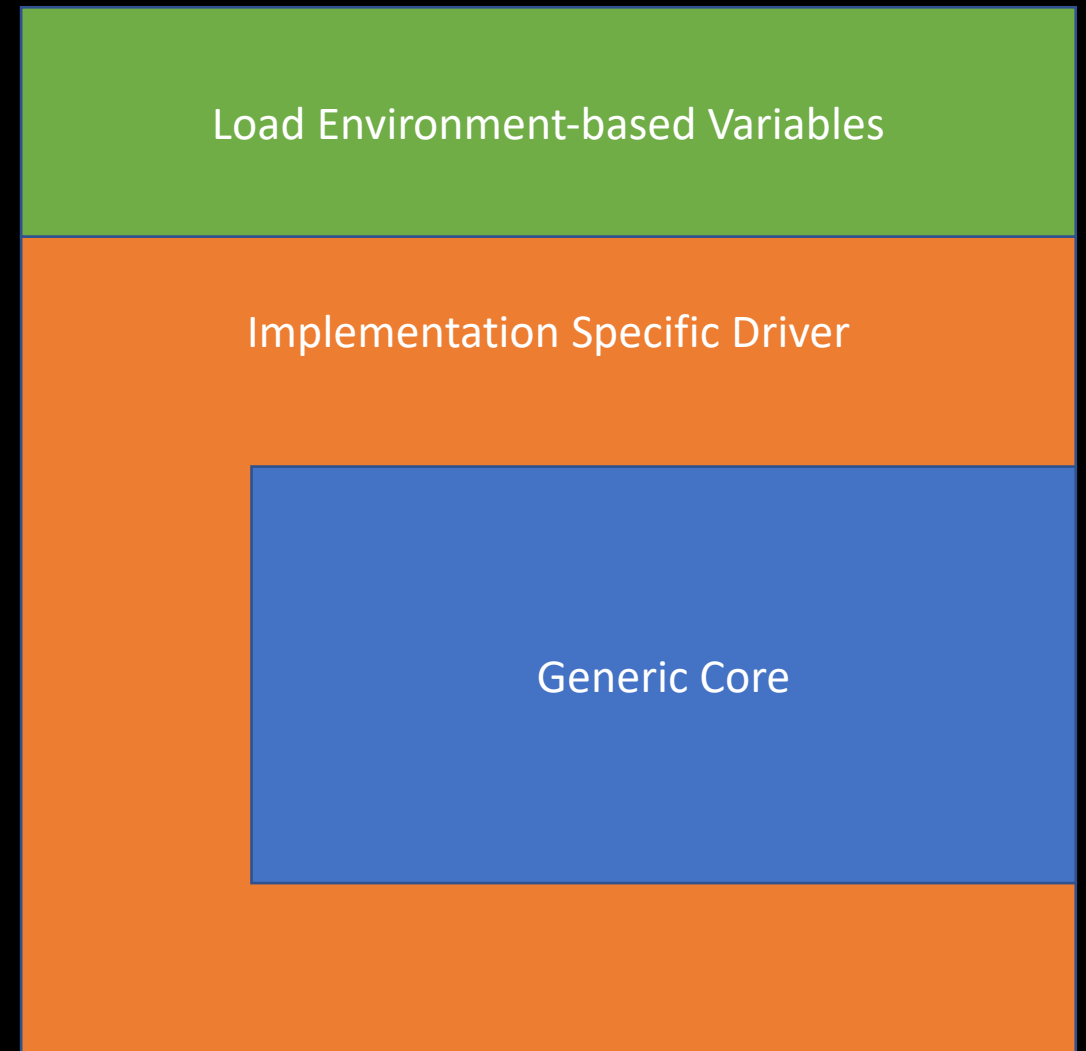
CONTRACTS

- Provisioning layer must create a compatible Ansible inventory
- Steady-state OS Layer may include sentinel state check for completion
- Most likely the steady-state OS layer should also contain common tasks for all ATC components such as SSL PKI creation and distribution (included sample in repository)

| | Responsibilities | Example Technologies | |
|-----------------------|--|---|-------------------|
| Application Layer | <ul style="list-style-type: none"> • ATC Components • Application Monitoring • Data Visualization | <ul style="list-style-type: none"> • Ansible push • Shell script | Execution Wrapper |
| Steady-state OS Layer | <ul style="list-style-type: none"> • OS Users/Groups • Package Repositories • Host-based Firewalls • Kernel Optimization | <ul style="list-style-type: none"> • Puppet • Chef • Salt • Ansible Tower • Ansible-pull | |
| Provisioning Layer | <ul style="list-style-type: none"> • DNS • Network • Compute • RAID | <ul style="list-style-type: none"> • Terraform • VinyIDNS • Foreman • MaaS | |

ATC COMPONENT ANSIBLE PLAYBOOK PATTERN

1. Load environment-based variables
2. Implementation-specific Pre-tasks
3. Generic Core role
4. Implementation-specific Post-Tasks



WEAVING EXECUTION ENVIRONMENT WITH GILT

SAMPLE GILT.YML:

```
- git: git@github.com:apache/trafficcontrol.git
  version: master
  files:
    - src: infrastructure/ansible/roles/ats
      dst: ../roles/ats
    - src: infrastructure/ansible/roles/dataset_loader
      dst: ../roles/dataset_loader
```

<https://github.com/metacloud/gilt>

VARIABLE PRECEDENCE

From most to least important

1. extra vars (always win precedence)
2. set_facts / registered vars
3. include_vars
4. include params
5. role (and include_role) params
6. task vars (only for the task)
7. block vars (only for tasks in block)
8. role vars (defined in role/vars/main.yml)
9. play vars_files
10. play vars_prompt
11. play vars
12. host facts
13. playbook host_vars/*
14. inventory host_vars/*
15. inventory file or script host vars
16. playbook group_vars/*
17. inventory group_vars/*
18. playbook group_vars/all
19. inventory group_vars/all
20. inventory file or script group vars
21. role defaults

◆ Used in lab



Photo by [Júnior Ferreira](#) on [Unsplash](#)

LAB MANAGER

GOALS

- Simple
- Focus on Data Relationships and Integrity
- Reliable System of Record
- Resolve inherent Chicken/Egg problem with ATC TrafficOps

CONCEPTS

- Environment definition & lifecycle
- Resource Pools
- Jobs
- Logs
- Fact Inventory



GraphQL API PROTOCOL



OPEN SOURCE PROTOCOL

Originally created by Facebook and donated to the Linux Foundation in 2017 where now it resides under the GraphQL Foundation.

Designed around flexibility of the client request. “Get what you want, only what you want, and nothing more.” Traditionally viewed as an upcoming alternative to REST.

<https://foundation.graphql.org>

Current adopters include:

- Facebook
- GitHub
- PayPal
- The New York Times
- Twitter

POSTGRESQL DATABASE



RELATIONAL DATABASE BACKEND

Originally created by engineers at UC Berkley with version 1 released in 1989, PostgreSQL continues to be a major force in Open-Source RDBMS.

<https://www.postgresql.org>

Current adopters include:

- Apache Traffic Control
- Uber
- Netflix
- Reddit
- Spotify

POSTGRAPHILE API



OPEN SOURCE GRAPHQL IMPLEMENTATION

Started in 2016, Postgraphile is an easy-to-use API library for GraphQL. The robust open-source NodeJS library is MIT licensed, however additional enterprise features are available for a small license fee.

Postgraphile is low to no-code required for a functional API as it leverages data from PostgreSQL to correctly build out the GraphQL Schema automatically with documentation that's available.

<https://www.graphile.org/postgraphile/>

While Postgraphile can be leveraged standalone or as a NodeJS library, I mix-in several other NodeJS libraries and frameworks for the Lab Manager:

- [ExpressJS](#)
- [JsonWebToken](#)
- [Grant](#)
- [GraphQL-Voyager](#)
- [Winston](#)



POSTGRAPHILE PRIMER

SECURITY

AUTHENTICATION

The Lab Manager leverages OAuth2.0 flows to obtain a valid JWT

ADAPTATION

The Lab Manager verifies the JWT and extracts the user, role, and capabilities to pass along through Postgraphile to PostgreSQL

AUTHORIZATION

Authorization is handled via native PostgreSQL security mechanisms built into the database.

SECURITY

NATIVE POSTGRESQL AUTHORIZATION

- Column
- Table
- Row Policies

ADDITIONAL INTEGRITY VALIDATION

- Usage of Check Constraints & Defaults to enforce JWT values

With the use of security definers, it is possible to override the security settings of a request and user

| Column Permissions | | | |
|--------------------|----------|----------|----------|
| | Column A | Column B | Column C |
| Row 1 | 1.A | | 1.C |
| Row 2 | 2.A | | 2.C |

| Table Permissions | | | |
|-------------------|----------|----------|----------|
| | Column A | Column B | Column C |
| Row 1 | | | |
| Row 2 | | | |

| Row Permissions | | | |
|-----------------|----------|----------|----------|
| | Column A | Column B | Column C |
| Row 1 | 1.A | 1.B | 1.C |
| Row 2 | 2.A | 2.B | 2.C |

| Check Constraint | | | |
|------------------|----------|----------|----------|
| | Column A | Column B | Column C |
| Row 1 | 1.A | User A | 1.C |
| Row 2 | 2.A | User B | 2.C |

BUSINESS LOGIC

GRAPHQL ISN'T JUST CRUD

Mutations in GraphQL vernacular encompass all potentially modifying operations.

```
mutation CreateMyDivision {
  createDivision(input:
    {division:
      {name: "MyDivision"}}
  ) {division {
    name
    nodeId
    regionsByDivision {
      nodes {
        name
      }
    }
  }}
}
```

BUSINESS LOGIC

GRAPHQL ISN'T JUST CRUD

Mutations in GraphQL vernacular encompass all potentially modifying operations.

```
mutation CreateMyDivision {
  createDivision(input:
    {division:
      {name: "MyDivision"}}
  ) {division {
    name
    nodeId
    regionsByDivision {
      nodes {
        name
      }
    }
  }}
}
```

```
mutation DeepDivisionCreation {
  deepDivisionCreation(input:
    {division:
      {name: "MyDivision"}
    },
    {region:[
      {name: "MyRegion1"}, {name: "MyRegion2"}
    ]}
  ) {division {
    name
    nodeId
    regionsByDivision {
      nodes {
        name
      }
    }
  }}
}
```

INTERESTED?

QUICKSTART BASE ENVIRONMENT

<https://github.com/apache/trafficcontrol/tree/master/experimental/graphql.sample>



SELF-SERVICE DATA CONCEPTS

ENVIRONMENT

BASIC FIELDS

- Name
- Description
- Owner
- Creation Timestamp
- Expiration Timestamp
- Type

COMPLEX FIELDS

- Gilt Configuration
- Configuration



RESOURCE POOLS

QUOTAS/AVAILABILITY

- What
- Where
- Constraints
- Support
- Type
- Identity
- Assignability

ASSOCIATIONS

- Environment
- Component
- CDN Delegation



JOB

LIFECYCLE

- Who
- Issuance Timestamp
- Last Update Timestamp
- Status
- Operation



LOGS

PROGRESS

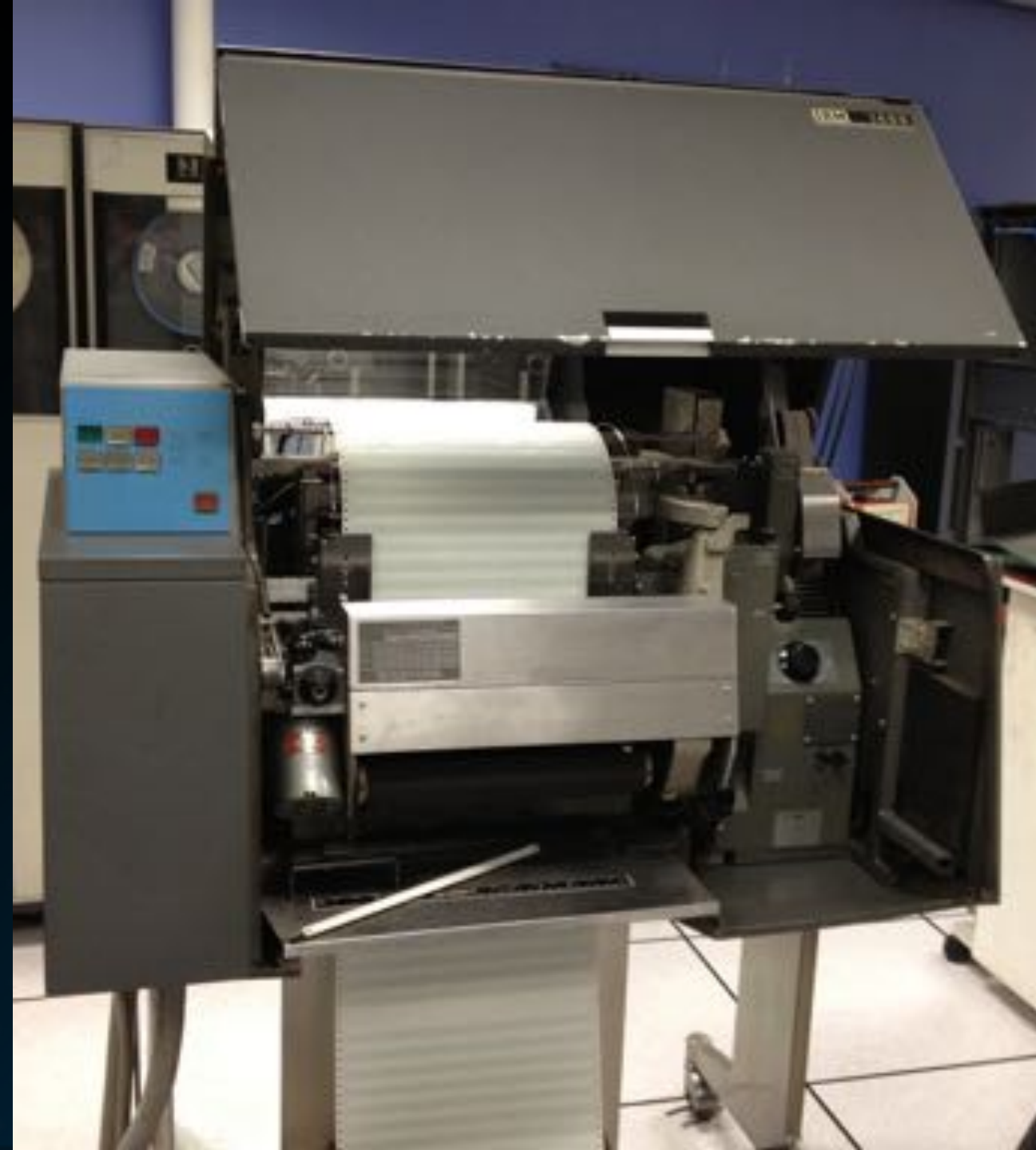
- Receipt Timestamp
- Job
- Message
- Classification

EXECUTOR

- Receipt Timestamp
- Job
- LogText

JOB

- Receipt Timestamp
- Job
- Playbook completion Timestamp
- Playbook File
- Playbook Line
- Task Name
- Task Parameters
- Task Payload
- Task Result
- Task Elapsed Time
- Task Target Host



FACT INVENTORY

CORE FIELDS

- Phase
- Payload
- Receipt Timestamp
- Environment
- CDN Component

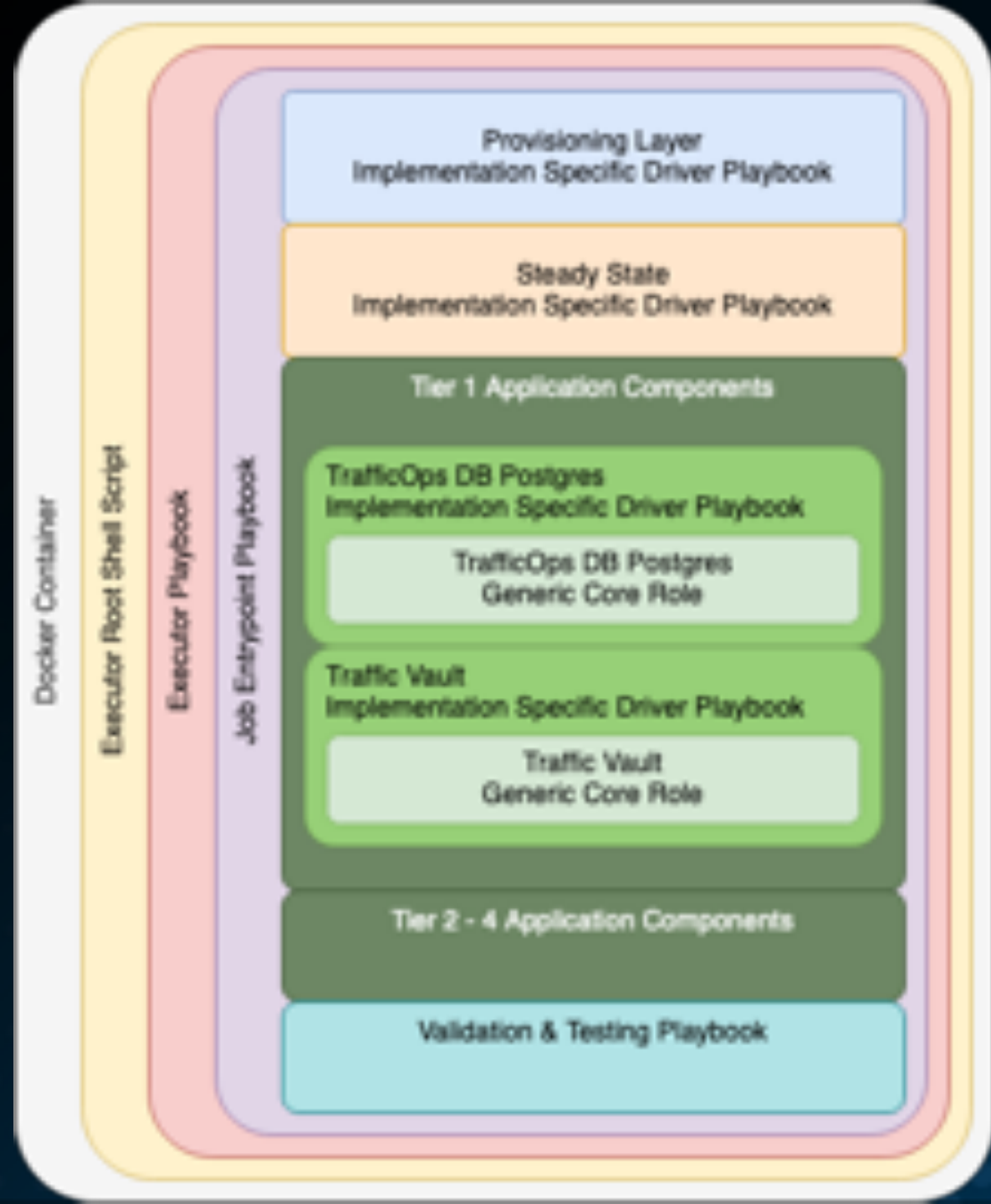
EXTRACTED FIELDS

- CPU
- Memory
- Disk
- FQDN
- NIC Types
- Manufacturer / Model



LAB EXECUTOR

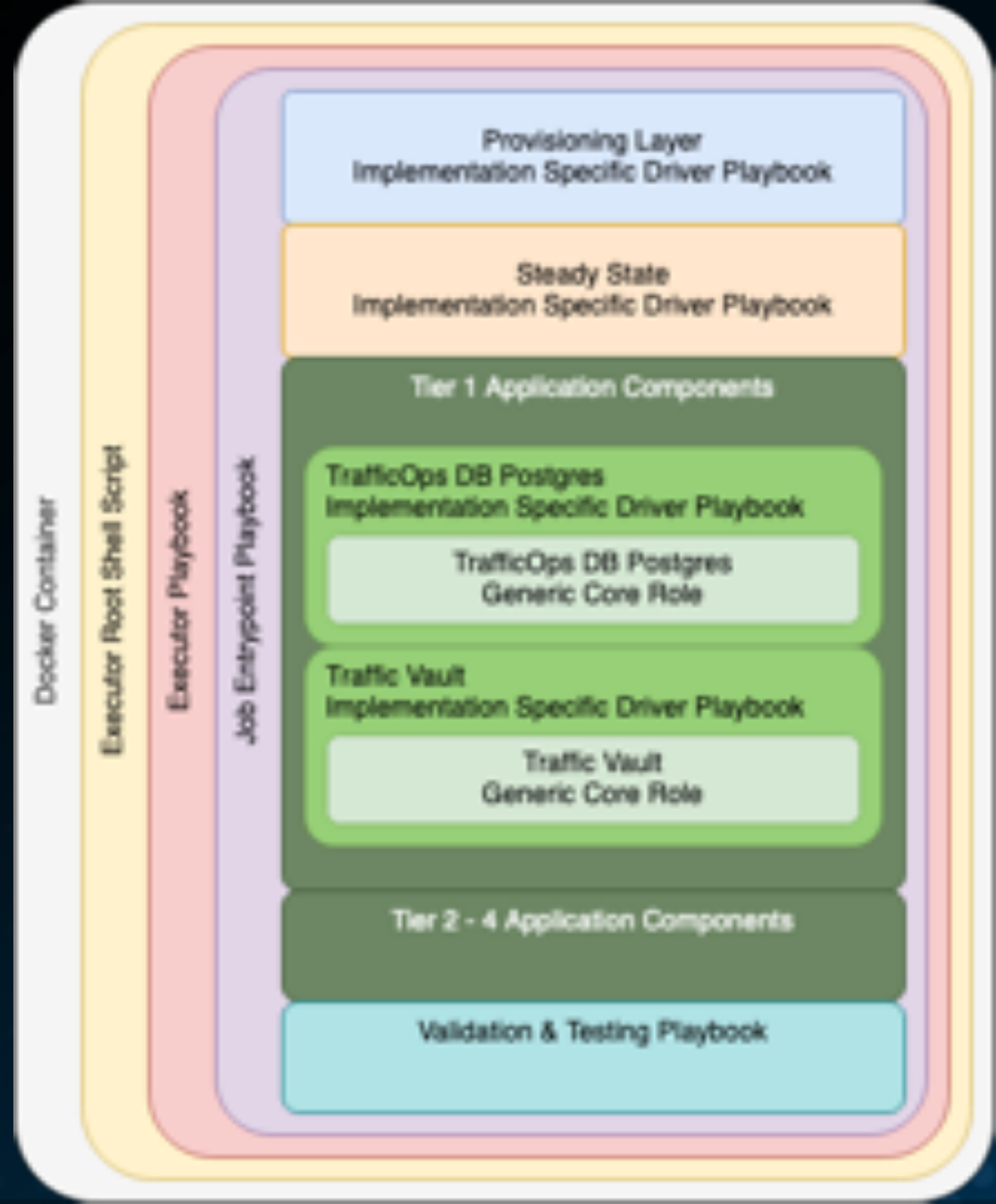
ABSTRACTIONS



ABSTRACTIONS

DOCKER CONTAINER

- Insulate Dependencies
- Improve Portability



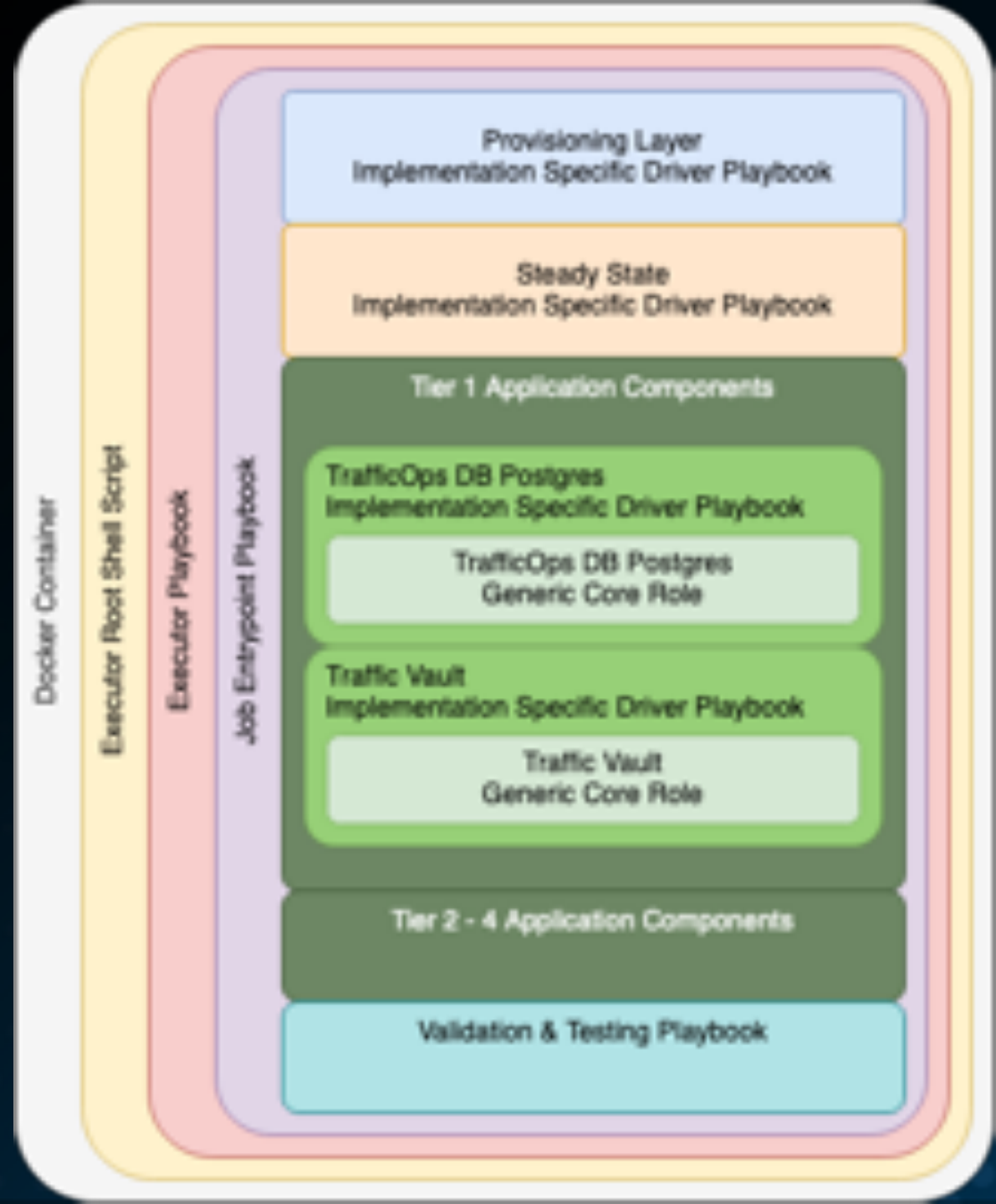
ABSTRACTIONS

DOCKER CONTAINER

- Insulate Dependencies
- Improve Portability

EXECUTOR ROOT SHELL SCRIPT

- Redirect its own output to itself
- Scrub & Submit Logs
- Update Job State



ABSTRACTIONS

DOCKER CONTAINER

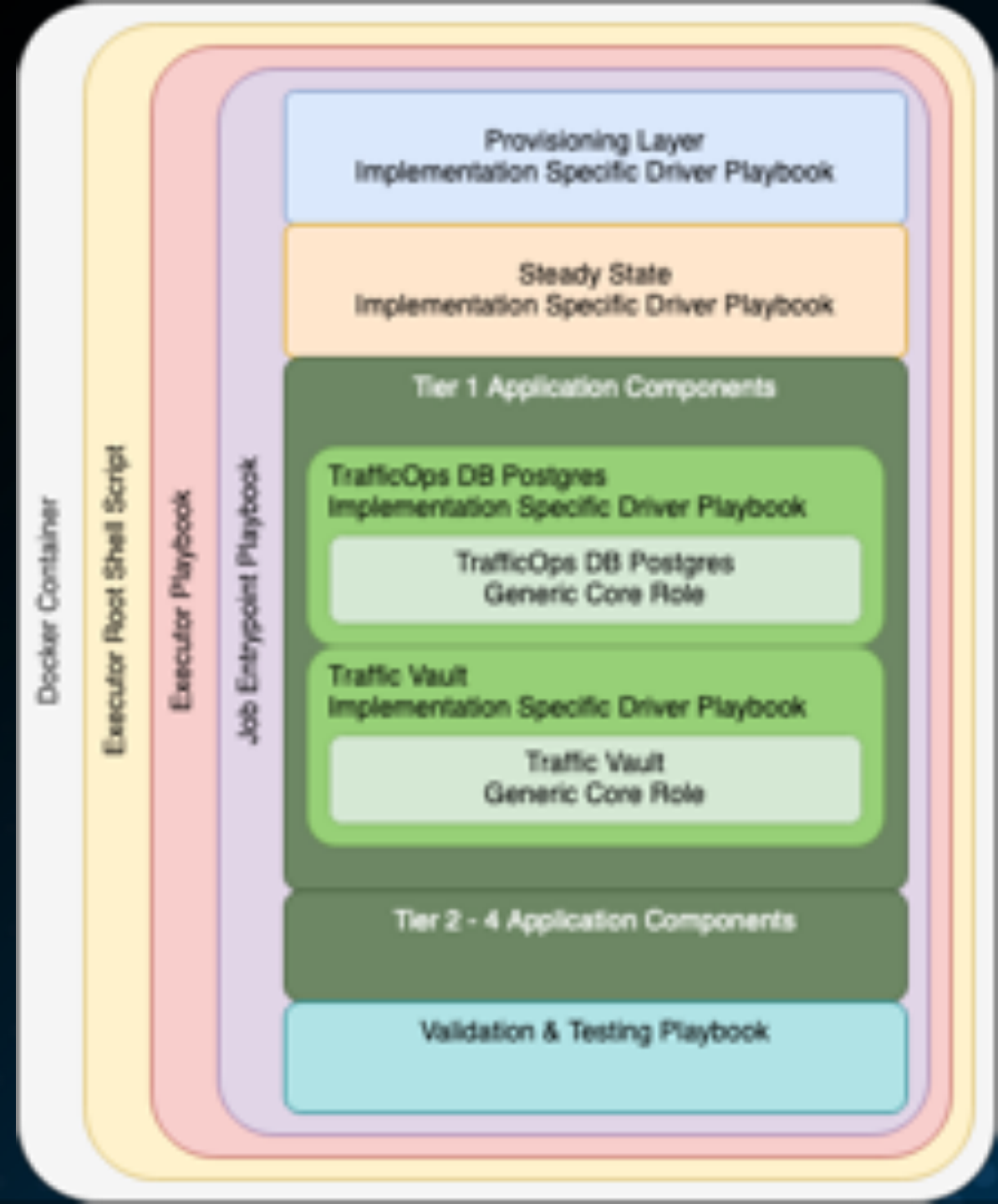
- Insulate Dependencies
- Improve Portability

EXECUTOR ROOT SHELL SCRIPT

- Redirect its own output to itself
- Scrub & Submit Logs
- Update Job State

EXECUTOR PLAYBOOK

- Obtain available Job
- Weave execution directory code
- Dump all job information



ABSTRACTIONS

DOCKER CONTAINER

- Insulate Dependencies
- Improve Portability

EXECUTOR ROOT SHELL SCRIPT

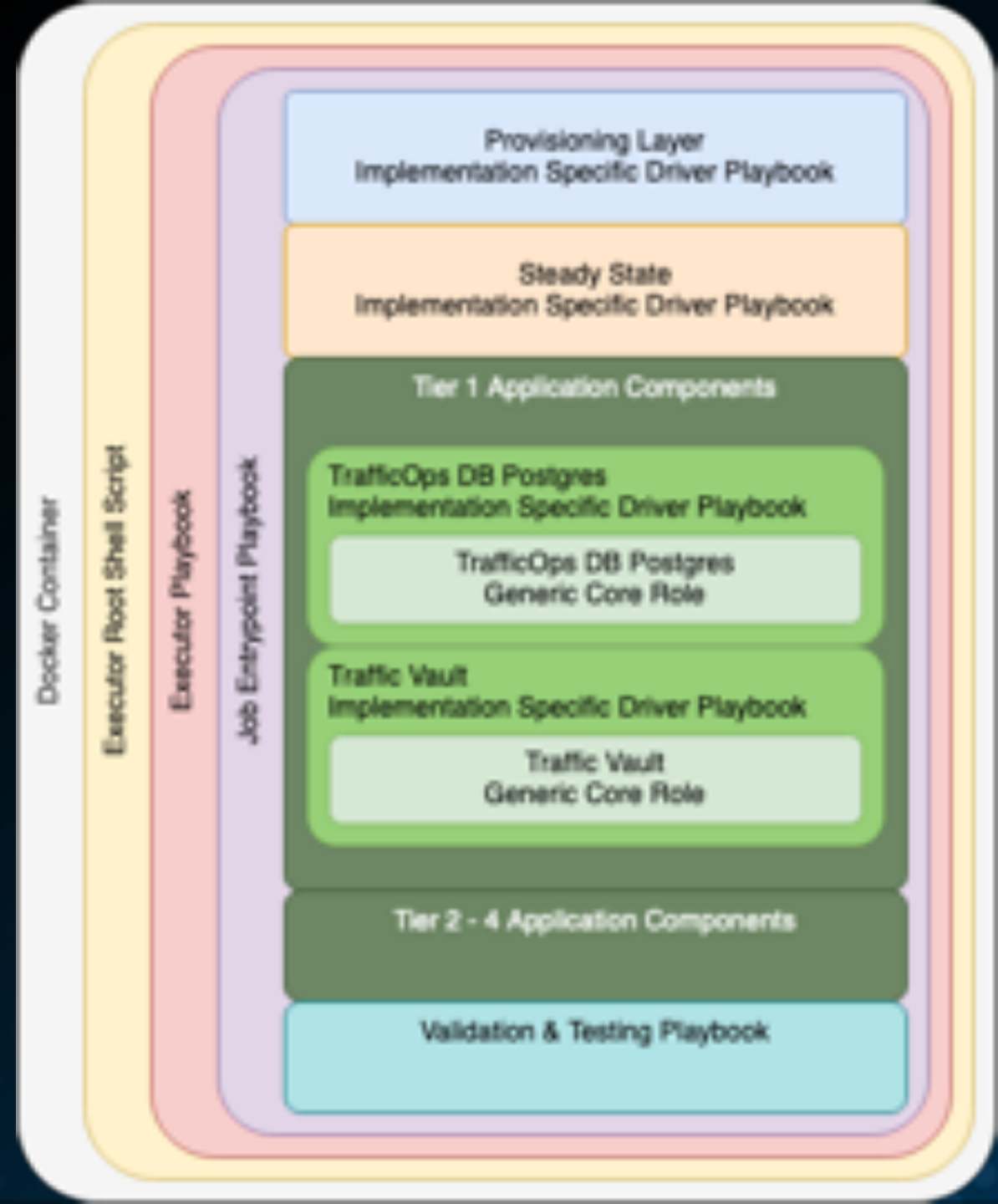
- Redirect its own output to itself
- Scrub & Submit Logs
- Update Job State

EXECUTOR PLAYBOOK

- Obtain available Job
- Weave execution directory code
- Dump all job information

JOB ENTRYPOINT PLAYBOOK

- Considered Main Execution for Job



EXECUTION LOGGING & SECURITY

ANSIBLE PARALLELIZATION

EXECUTE & POLL

```
- name: My long running task
  some_module_name:
    module_arg1: "{{ item }}"
  loop: "{{ large_array }}"
  register: long_task_result
  async: 60
  poll: 5
```

- Every 5 seconds check if all the parallel tasks are done for up to 60 seconds

ANSIBLE PARALLELIZATION

EXECUTE & POLL

```
- name: My long running task
  some_module_name:
    module_arg1: "{{ item }}"
  loop: "{{ large_array }}"
  register: long_task_result
  async: 60
  poll: 5
```

- Every 5 seconds check if all the parallel tasks are done for up to 60 seconds

FIRE & FORGET

```
- name: My long running task
  some_module_name:
    module_arg1: "{{ item }}"
  loop: "{{ large_array }}"
  register: long_task_result
  async: 60
  poll: 0
```

- Launch the async task which can run up to 60 seconds in total while continuing execution.

ANSIBLE PARALLELIZATION

FIRE & REVISIT

```
- name: My long running task
  some_module_name:
    module_arg1: "{{ item }}"
  loop: "{{ large_array }}"
  register: long_task_result
  async: 60
  poll: 0
```

- Launch the async task which can run up to 60 seconds in total while continuing execution.

```
- name: Wait for long running task
  async_status:
    id: "{{ item.ansible_job_id }}"
  loop: "{{ long_task_result.results }}"
  register: long_task_polling_result
  retries: 20
  delay: 2
  until: long_task_polling_result.finished
```

- Poll up to 20 times with 2 seconds between for all forks of the async task to finish

METATEMPLATING

METATEMPLATE TASK

```
- template:  
  src: post_tmpl.json  
  dest: post_vars.json  
  block_start_string: "{*"  
  block_end_string: "*}"  
  variable_start_string: "{@"  
  variable_end_string: "@}"
```

EXAMPLE

```
"BaseFQDN": "{@ common.env @}.kabletown.invalid"  
"target_version": "{{ production_version }}"  
"to_version": "{{ target_version | default(omit) }}"
```



FUTURE WORK

WHAT'S COMING

REFACTOR PR

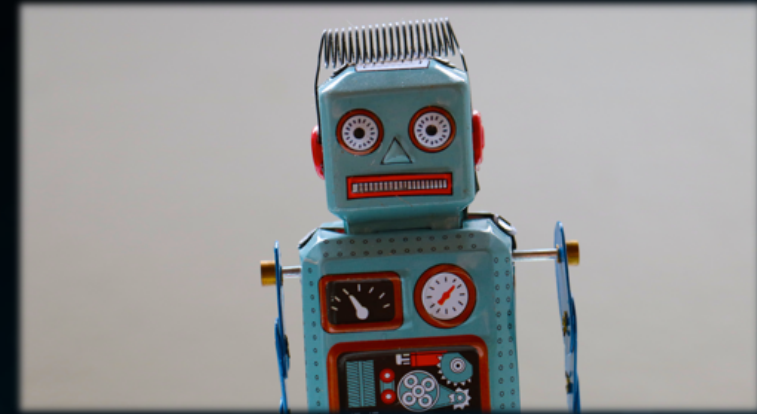
<https://github.com/jhg03a/trafficcontrol/tree/ansible.refactor>

- ATS
 - Better cleanup of previous ATS cache data
- Dataset Loader
 - Better retry logic
 - Support for hardware variations in ATS profile templates
 - Per component server object defaults
- TR
 - Better log rotation cleanup
- TO
 - Better failure logic with Postinstall & admin
- TODB
 - Support for up to 3 tier replication topology





TAKEAWAYS

1. See how Comcast has leveraged the Open-Source Ansible roles for ATC.
2. Learn more about technology stack choices we've made.
3. Gain a better understanding of how deep the rabbit hole goes with modeling complex systems.
4. Inspire you to think about your own movements toward self-service.



Jonathan Gray

 Twitter : @jhg03a

 GitHub : @jhg03a

 traffic-control-cdn.slack.com : @jhg03a

 jhg03a@apache.org

<https://unsplash.com/photos/R4WCbazzrD1g>
https://about.twitter.com/en_us/company/brand-resources.html
<https://github.com/logos>
<https://slack.com/media-kit>
https://commons.wikimedia.org/wiki/File:Antu_mail-folder-sent.svg