EVERY APP
ANYWHERE

Super NetOps Program
Automation & Orchestration

Charles Chong Solution Architect, APCJ

赞助商：

F5支持公益：

自动化和编排概念

Automation & Orchestration Concepts

有经济效益的自动化系统需要**合适的简略架构**

Cost effective automation requires appropriate Abstraction

什么是合适的简略架构呢？

What does it mean by appropriate Abstraction?

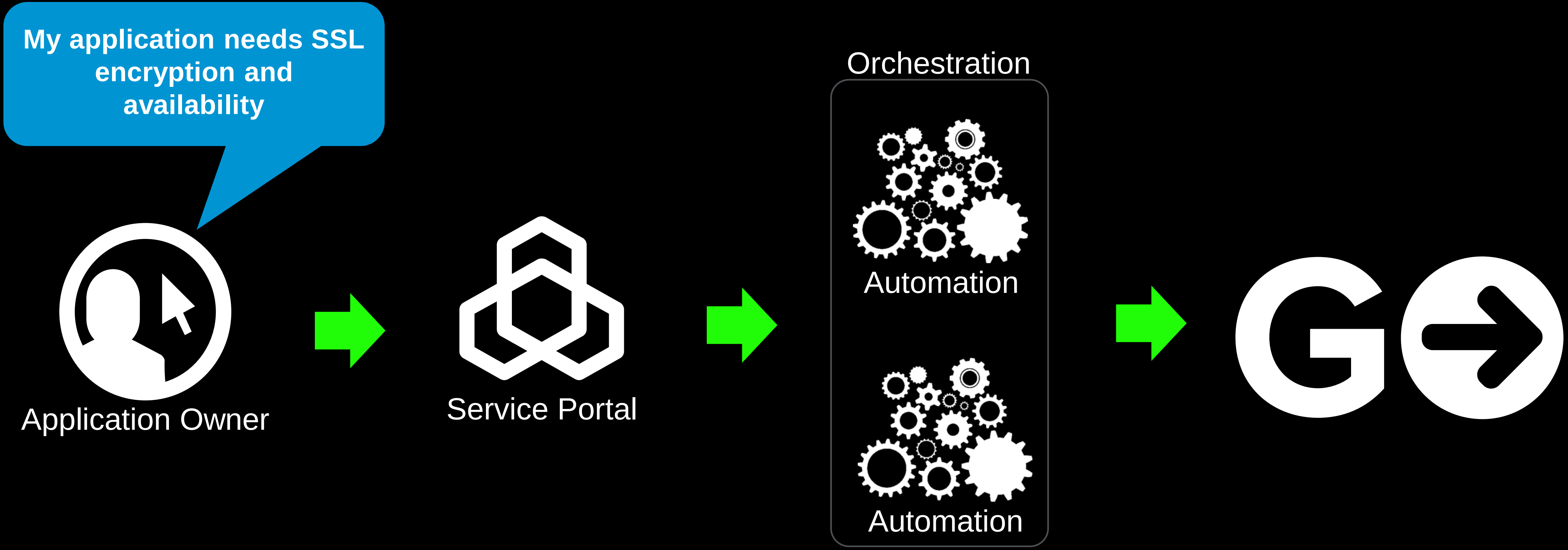
架构简略 - 是隐藏繁复应用系统的技巧之一。透过此技巧，用户可以用简单的方式与系统沟通，以达致 “深入浅出” 的效果，将繁复的程序抑制在底层

Abstraction is a technique for hiding complexity of **computer** systems. It works by establishing a level of simplicity on which a person interacts with the system, suppressing the more complex details below the current level

有经济效益的自动化系统
需要**合适的简略架构**

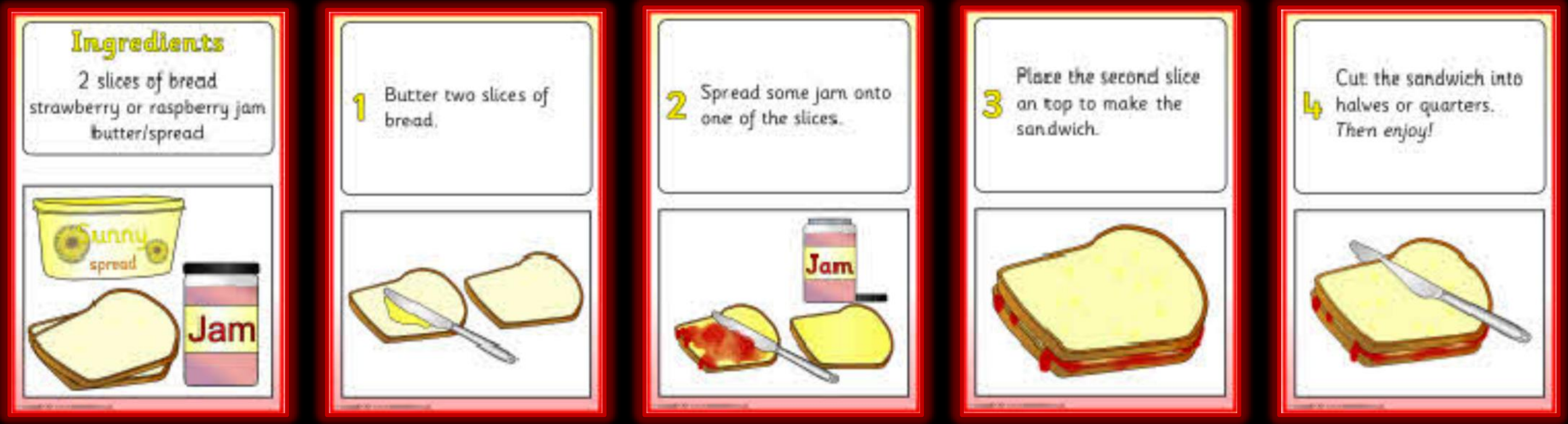
有了合适的简略架构，
就可以有**宣告式的介面和编程模式**

Appropriate abstraction enables Declarative Interfaces



Imperative Model 命令式编程模式

Imperative – What we’ve done for years (scripting, iRules, etc.) Imperative methodology implies that you define the flow of an operation implicitly. It also implies that domain-specific knowledge is required to interact with the system.



Ingredients
2 slices of bread
strawberry or raspberry jam
butter/spread

1 Butter two slices of bread.

2 Spread some jam onto one of the slices.

3 Place the second slice on top to make the sandwich.

4 Cut the sandwich into halves or quarters. *Then enjoy!*

What domain-specific knowledge is required to make this sandwich?

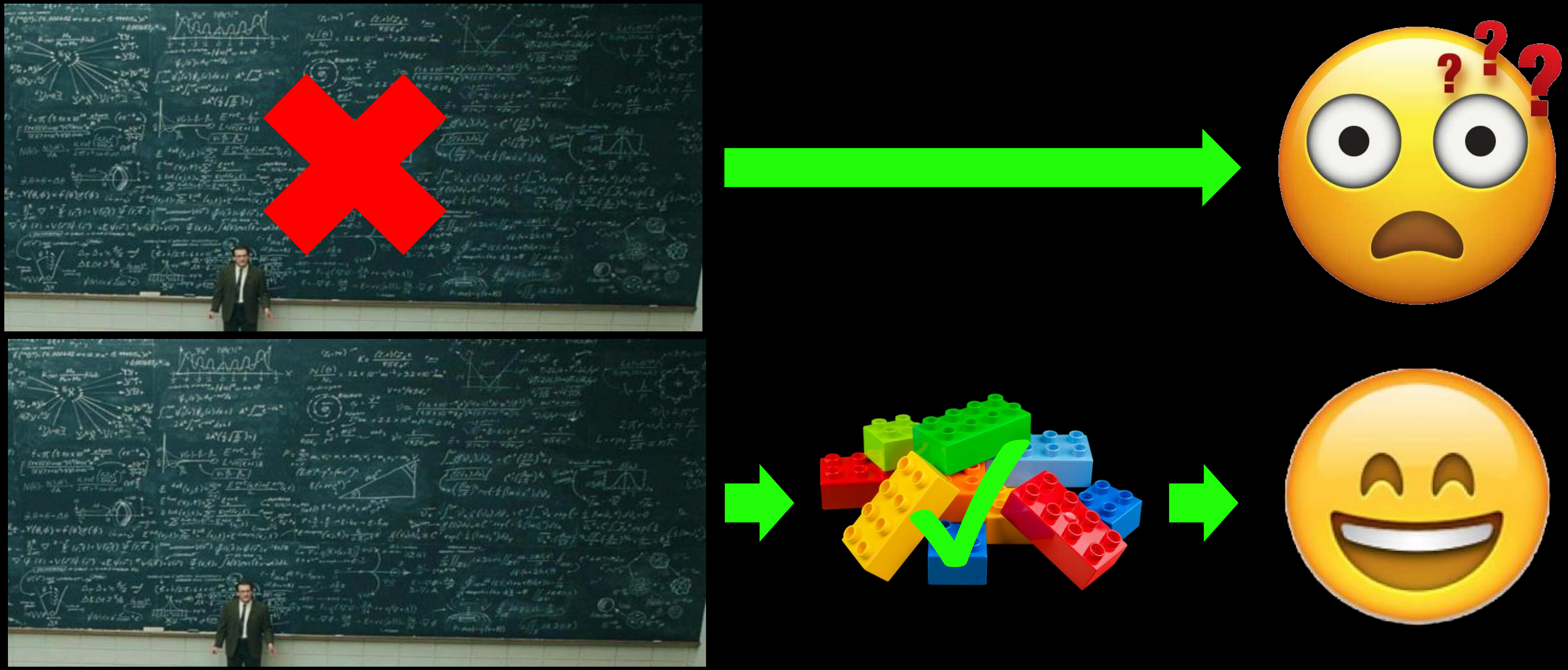
Declarative Model 宣告式编程模式

- 宣告式方式 – 这将是大趋势, 会被广泛使用
- Declarative methodology 能简单的应用底层的系统运作, 为我们想要的结果，来作出具体的定义 。
- 好处是简化或直接挪去所需的领域知识

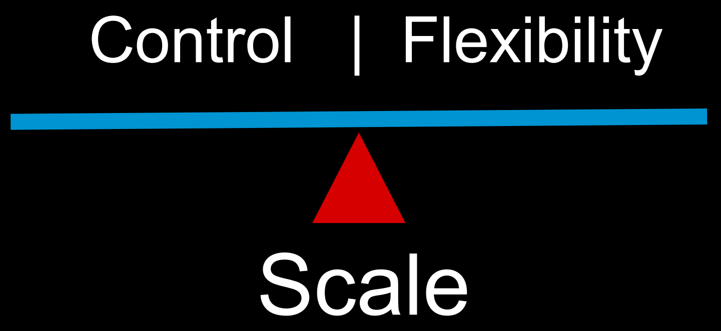
Declarative – What we're evolving to. Declarative methodology implies that you define the desired outcome and depend on underlying mechanisms to deliver that outcome. This methodology tries to reduce or eliminate the need for domain specific knowledge.

宣告式的编程模式能减少或去除**所需的领域知识**

Declarative Interfaces reduce or eliminate Domain Specific Knowledge



Practical Example - Role Based Access Control



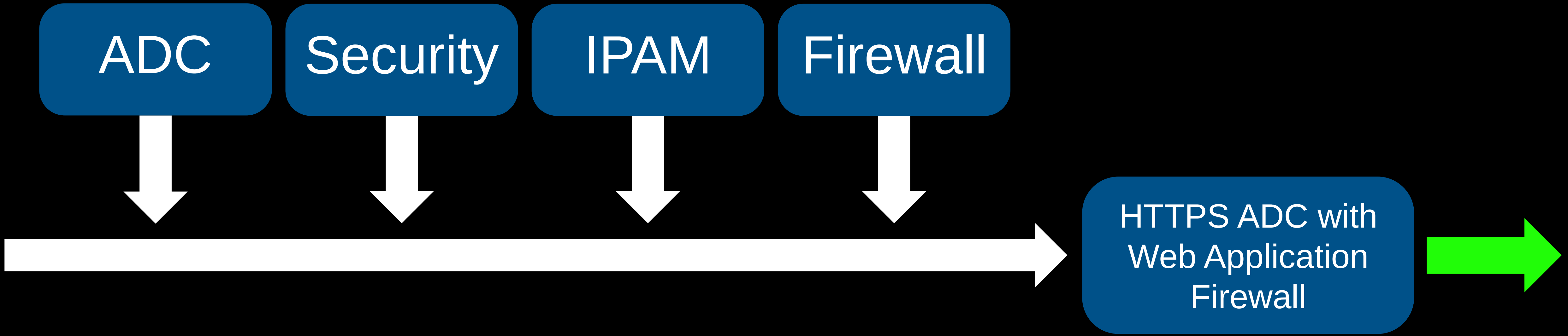
- **Imperative Methodology 命令式方式**– 针对环境里的每项设备进行**访问控制**设置，以确保系统安全，并预防设置疏忽而影响系统运作
- **Declarative Methodology 宣告式方式**– 去除设备层的政策设置。 以宣告式的中央介面为用户提供了一套安全，零疏忽的设置管理。用户只能在自个儿的权限里进行设置管理
- 能以更大的规模有效地管制设置

减去了所需的领域知识，
能使 Super NetOps 和 DevOps
有效率地彼此合作，
让应用程式更快速和更完善地推广到市场

Reduced requirement for Domain Specific Knowledge enables effective collaboration between Super NetOps and DevOps

少了所需的领域知识，我们便能够简化 DevOps 对不同设备所需的自动化与编排过程

Lower domain specific knowledge enables simplified DevOps Orchestration



简化对不同设备的自动化与编排能促进高效率的DevOps文化

Orchestration enables an effective DevOps culture

Concepts – Multi-Tenancy多租户概念

- BIG-IP is Multi-Tenant inherently.
- Route-Domains
- Partition 分区·
- What about route-domain 0?

- BIGIQ implements a Provider/Tenant model for multi-tenancy.

- All automation code should be multi-tenant aware by nature.
- It’s not that hard… always use a partition for object names.
- Always use a route domain on all L3 addresses (including 0).

Concepts – Source of Truth(tiness) 设置来源的真理性

- **Source of Truth** – 应用服务的设置命令必须是单一来源并具有权威性的
- Changes for a service should propagate (push) from the source of truth to sub-ordinate systems. 所有的设置更动必须由特定的主要源头传播 / 推广至属下的系统
- 必须妥善处理源头以外的设置更动（最好避免出带的设置）
- To address real-world challenges we prefer to call this 'Source-of-Truthiness':
- In an automation tool chain it's possible to create layers of 'truth' as long as sub-ordinate systems 'filter' the service definition upstream.
- What the northbound system doesn't know won't hurt it.
- 设置更动务必由北向下传至南向的属下系统。必须严防设备上传设置指示。
- iApp strict updates 是 Source Of Truthiness 严谨监控设置更动的最佳典例
- BIGIQ service catalog 或 服务目录 是 Source-of-Truthiness 的范例.

Concepts – F5 松绑式的自动化架构

REST API,
iApps,
iWF Service Catalog
Python SDK,
PyControl,
TMUI, TMSH

LAYER 4-7

STATEFUL SERVICES

ADC

Local Load Balancing

Application Security

Application Performance

Secure Web Gateway

Global Load Balancing

DDoS Protection

Identity and Access

Application Proxies

Firewall

LOOSELY COUPLED 松绑式架构

iWF Connectors,
Device Onboarding,
Cisco Device Pkg, LBaaS

LAYER 2-4

STATELESS SERVICES

Router

Switch

Automating F5

自动化F5设备与功能

主要的3大工具 - 3 Key Automation Tools

- REST API
- iApp Templates & iApp App Services
- BIG IQ

Introduction to REST

- Based on HTTP and JSON
- Uses HTTP verbs (GET, POST, PUT, PATCH, DELETE)
- Data is sent using the Javascript Object Notation format

-
  ```
  {
    “attribute1”:”value1”,
    “attribute2”:[“array”,”of”,”values”],
    “attribute3”: [ { “nested1”:”value1”, “nested2”:”value2” }, {“nested3”:”value3”} ]
  }
  ```

REST APIs and HTTP Verbs

- The following table summarizes the interpretation of HTTP methods for REST APIs.
- HTTP methods (verbs) are used to create, read, update, and delete (CRUD) resources.
- APIs must use HTTP verbs in a manner described in the table below.
- APIs may implement a subset as required.

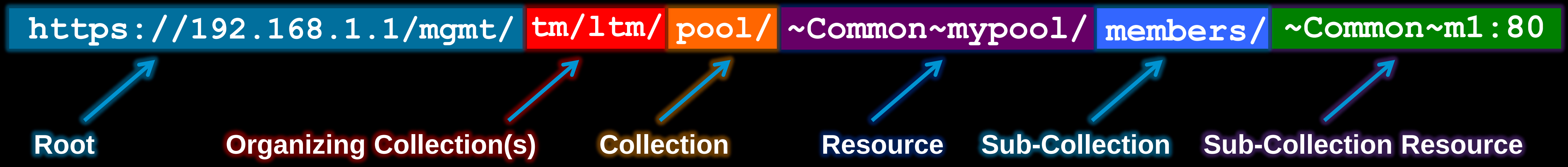
URI	POST	GET	PUT	DELETE	PATCH
Collection	Create resources.	Get representation of all resources in the collection.	Fully update all resources in a collection.	Delete all resources in a collection.	Partially update all resources in a collection.
Resource	Used for non-idempotent controller resources.	Get a resource’s representation.	Fully update the resource if it exists.	Delete a resource.	Partially update a resource.

Response Codes

- APIs must make use of HTTP response codes where appropriate.
- The following table describes the required success response codes.

Response Code	Applicable Verbs	Notes
200 OK	• All	Return on most positive responses including DELETE.
201 Created	• POST	HTTP Location header contains link to newly created resource.
202 Accepted	• POST • PUT • PATCH • DELETE	Return when a request will take a long time; server should return a Location header for client to get state updates.
404	• GET	The resource does not exist.
500	• All	Check /var/log/restjavad.0.log

Anatomy of a REST URI

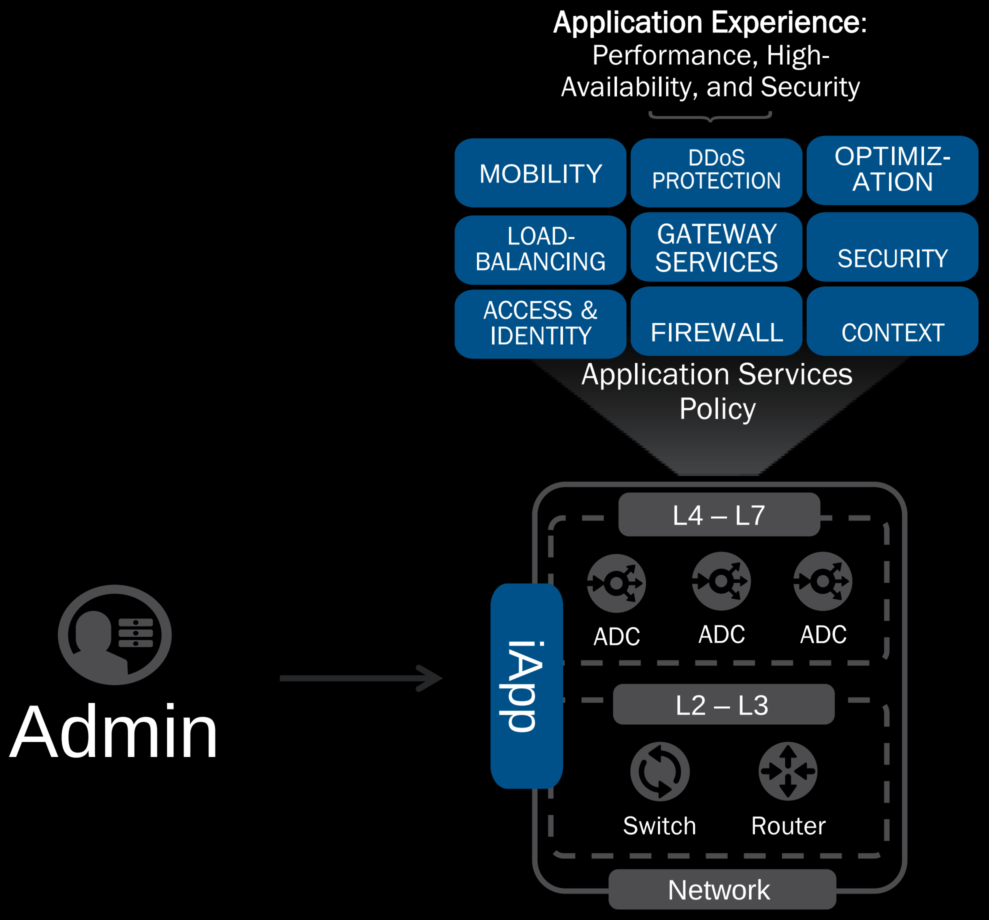


```
https://192.168.1.1/mgmt/tm/ltm/pool/~Common~mypool/members/~Common~m1:80
```

NOTE: Resource names map ‘~’ to ‘/’ (e.g. ~Common~mypool is really /Common/mypool)

iApps – 进行简化设置的工具

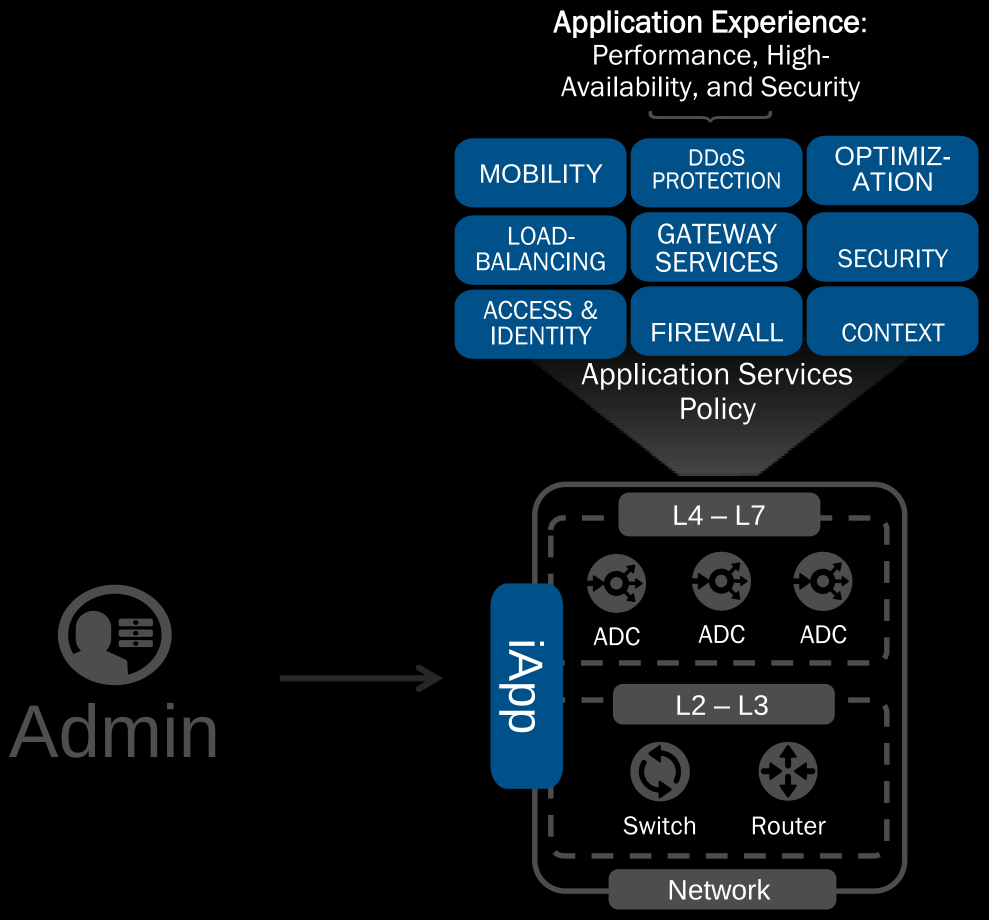
- 简化 L4-7 的设置 **Config** Abstraction
- 全面支持平台所拥有的功能
- 能针对特定应用作出定制
- 能按用户的任务 / 角色显示不同的设置介面 – ie. Network Team 和 Application Team



- L4-7 **Config** Abstraction
- Full platform capability
- Site-specific customization
- User – ie. Network Engineer

iApp 模版 (Templates)

- 简化底层平台繁复的应用服务功能呼叫程序
- 桥接命令式 **(Imperative)**和宣告式**(Declarative)**的编程模式
- 应用服务部署只需一个API Call
- 支持定制设置 Enable granular access to configuration (as needed)



App Services iApp

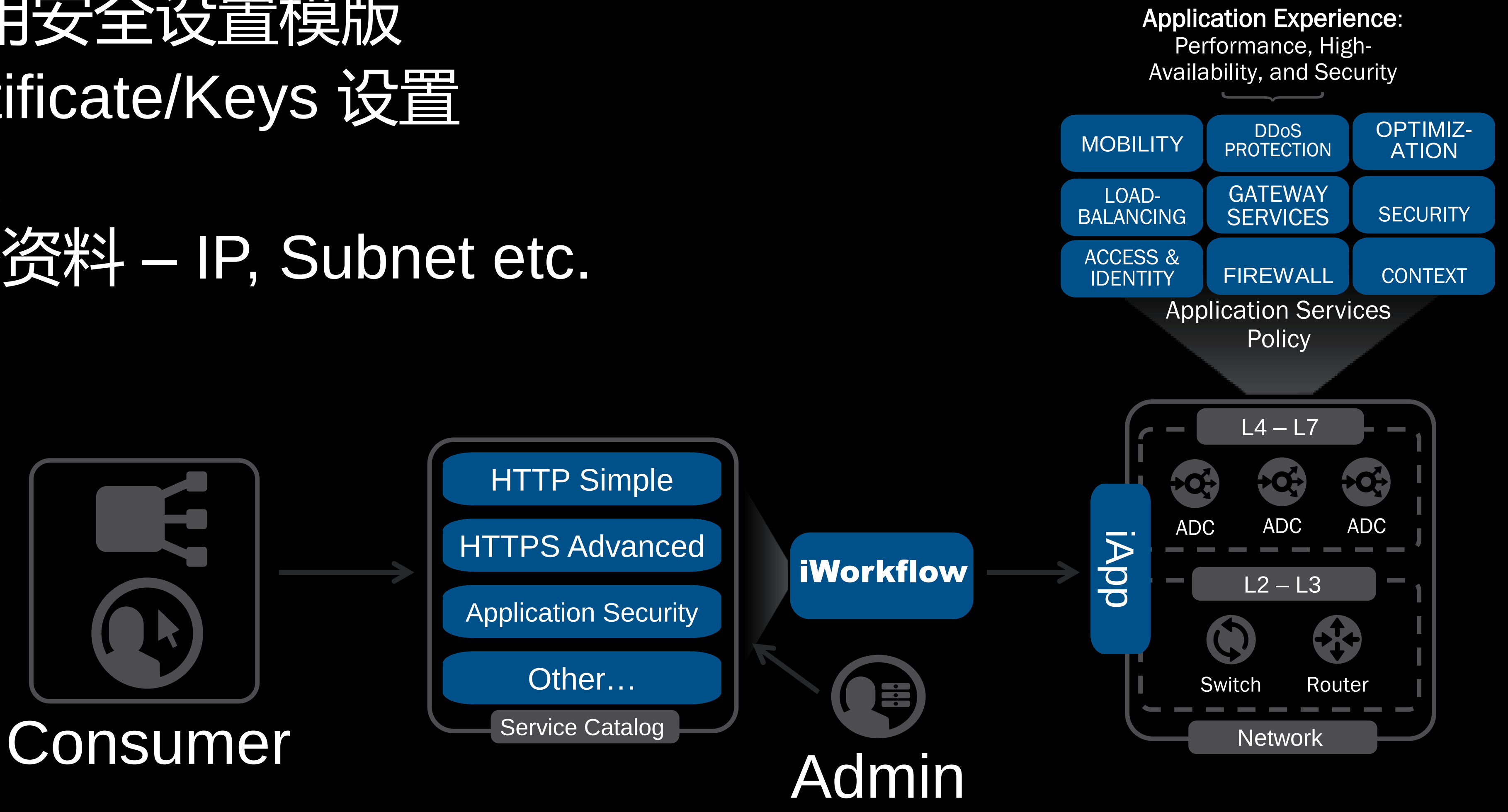
- Designed specifically to enable Abstraction & Automation
- Traditional iApp templates focused on a ‘wizard’ based UI interface, not automation
- Fully supported by F5 Support Team
- Implements an **Eventually Consistent** deployment model
- Compatible with wide range of F5 TMOS software versions
- Covers many L4-7 use cases including Security Services
- Developed and released on GitHub
- https://github.com/F5Networks/f5-application-services-integration-iApp

最终一致的部署 (Eventually Consistent Deployment)

- L4-7 Service Deployments can involve multiple steps
- **服务部署是一项异步作业 -** Service Deployment should be an Asynchronous operation
- 用户以咨询的方式来确认部署的成与败 - User queries the status of the deployment to determine success or failure
- 无需依次序等候，自动化的作业流程将不受影响，能继续毫无间断地完成 - Allows other processes in the automation change to continue

BIG IQ – 提供宣告式介面 Declarative Interface

- L4-7 **Service** Abstraction
- Tenant/Provider 应用服务模式
- Eg. 设定应用安全设置模版
  - SSL Certificate/Keys 设置
  - 安全政策
  - 详细网络资料 – IP, Subnet etc.

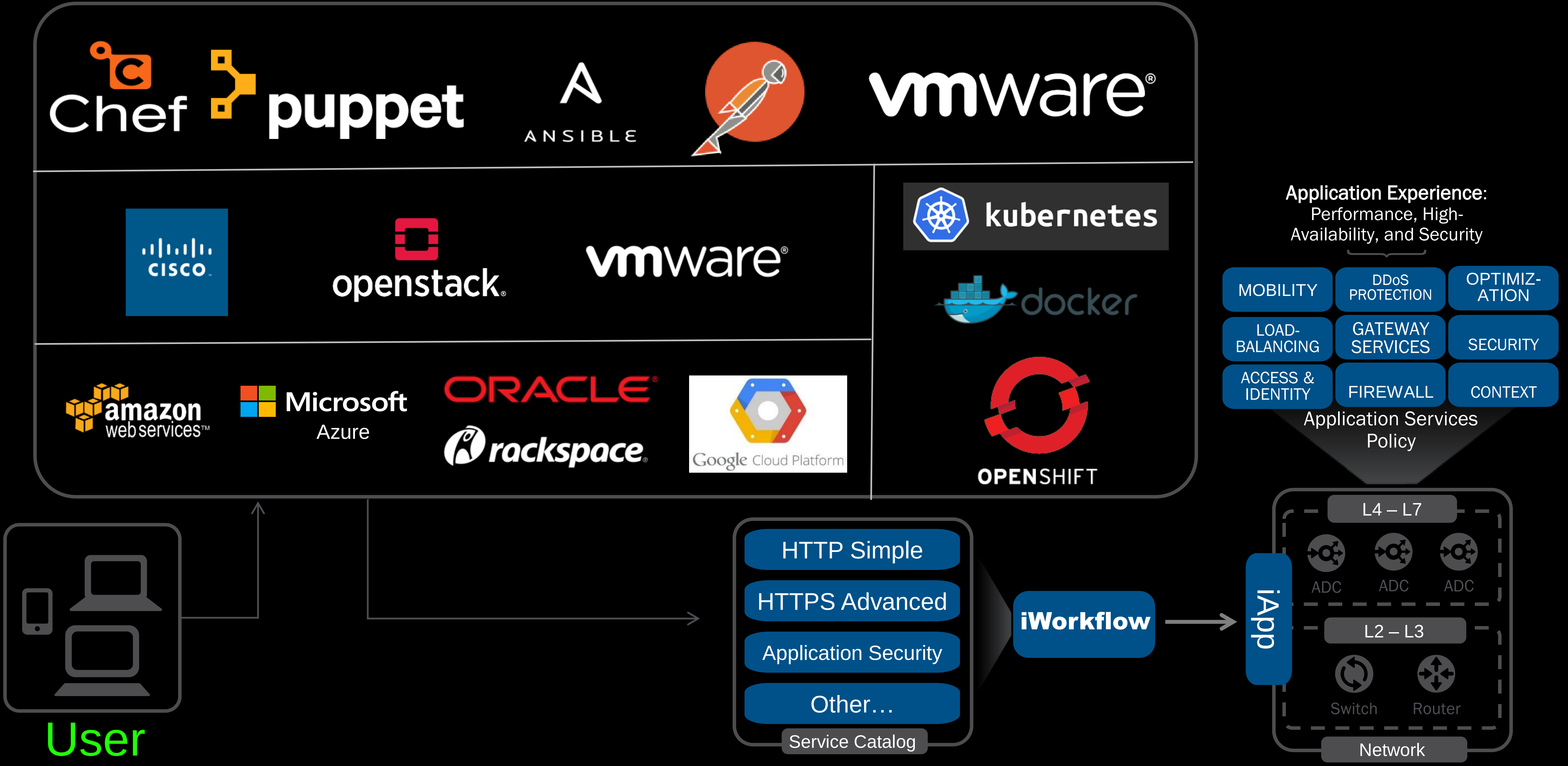


工具链组合 Toolchain Components

- **App Services iApp**
- 简化自动化设置并把BIG-IP 功能服务化 (Abstract BIG-IP configuration into Services)
- BIG-IP 的应用服务可透过简单的REST设定 / 默认设置完成所需的编排流程

- **BIGIQ**
- 透过中央系统，集中所有的应用服务和相应设备，让用户使用简单的介面统一应用服务的管理和部署

宣告式自动编排 Declarative Orchestration/DevOps



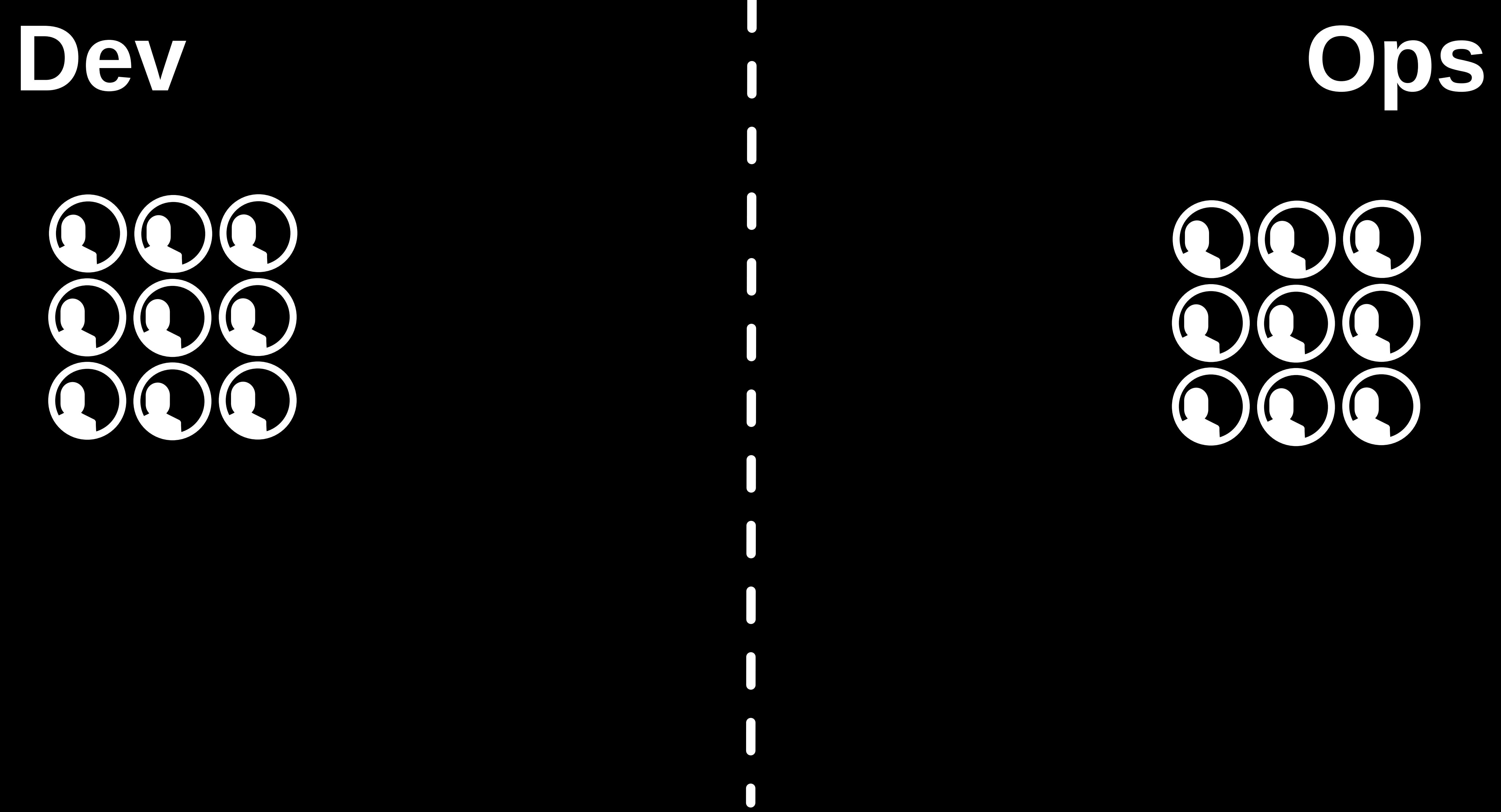
Continuous Integration

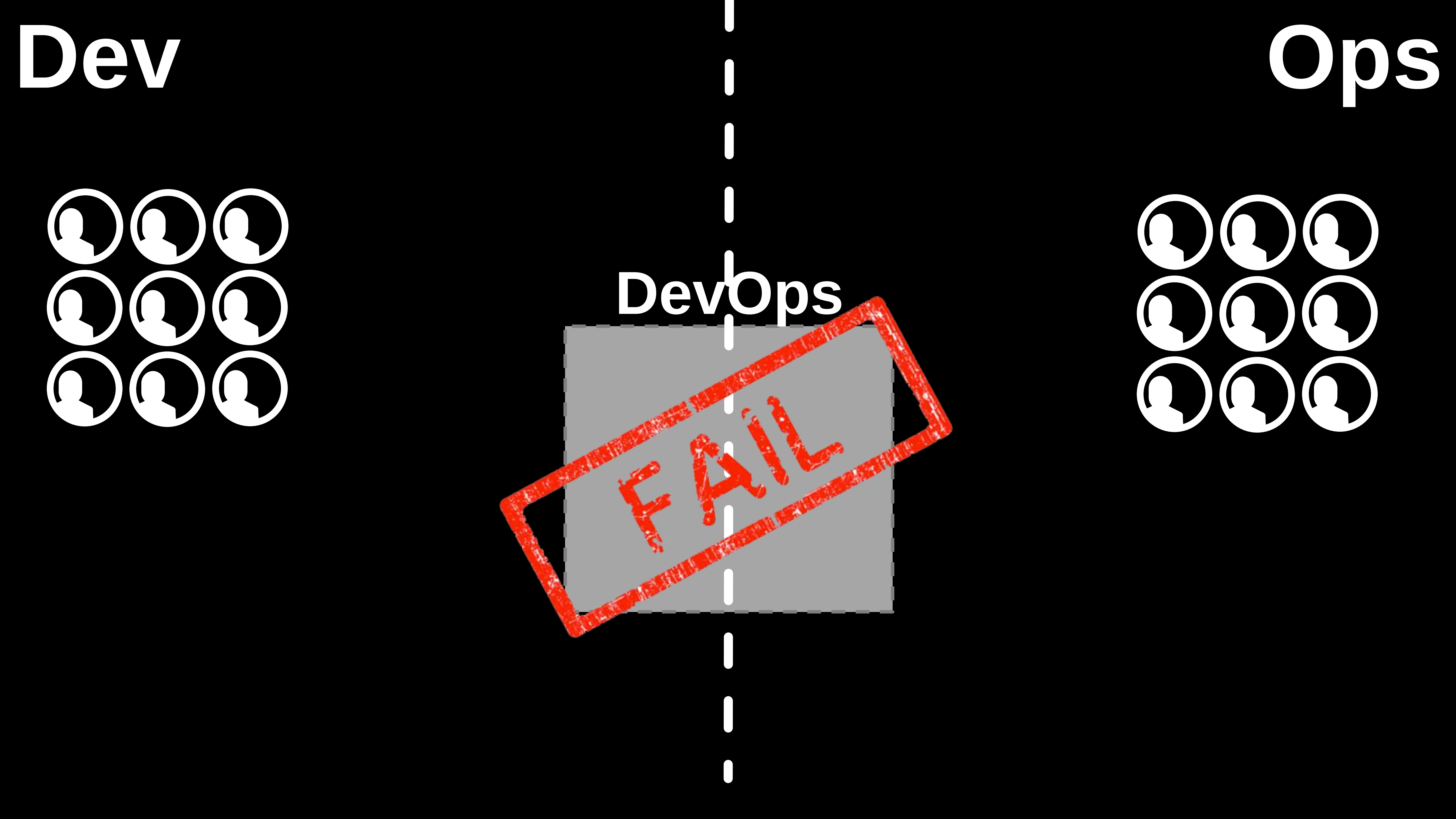
Continuous Development

CI/CD

Dev

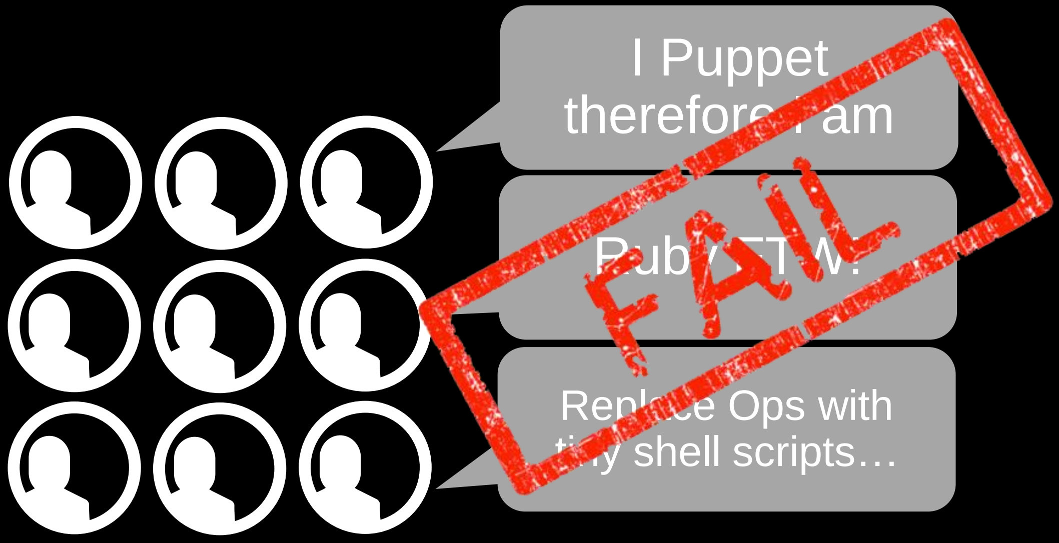
Ops

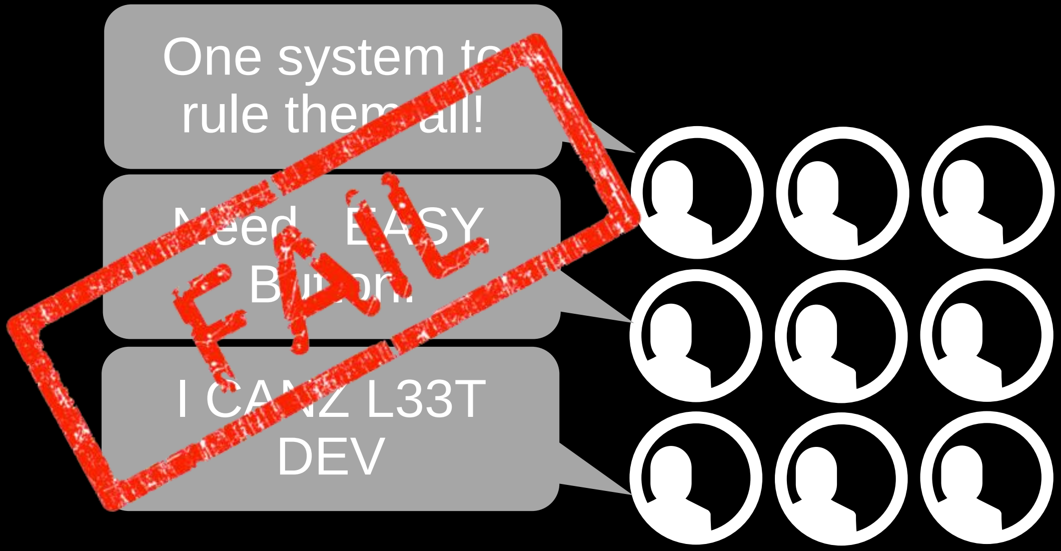



Dev
Ops
DevOps
FAIL

Dev

Ops


I Puppet therefore I am
Ruby FTW!
Replace Ops with tiny shell scripts…
FAIL


One system to rule them all!
Need EASY Button!
I CANZ L33T DEV
FAIL

Dev

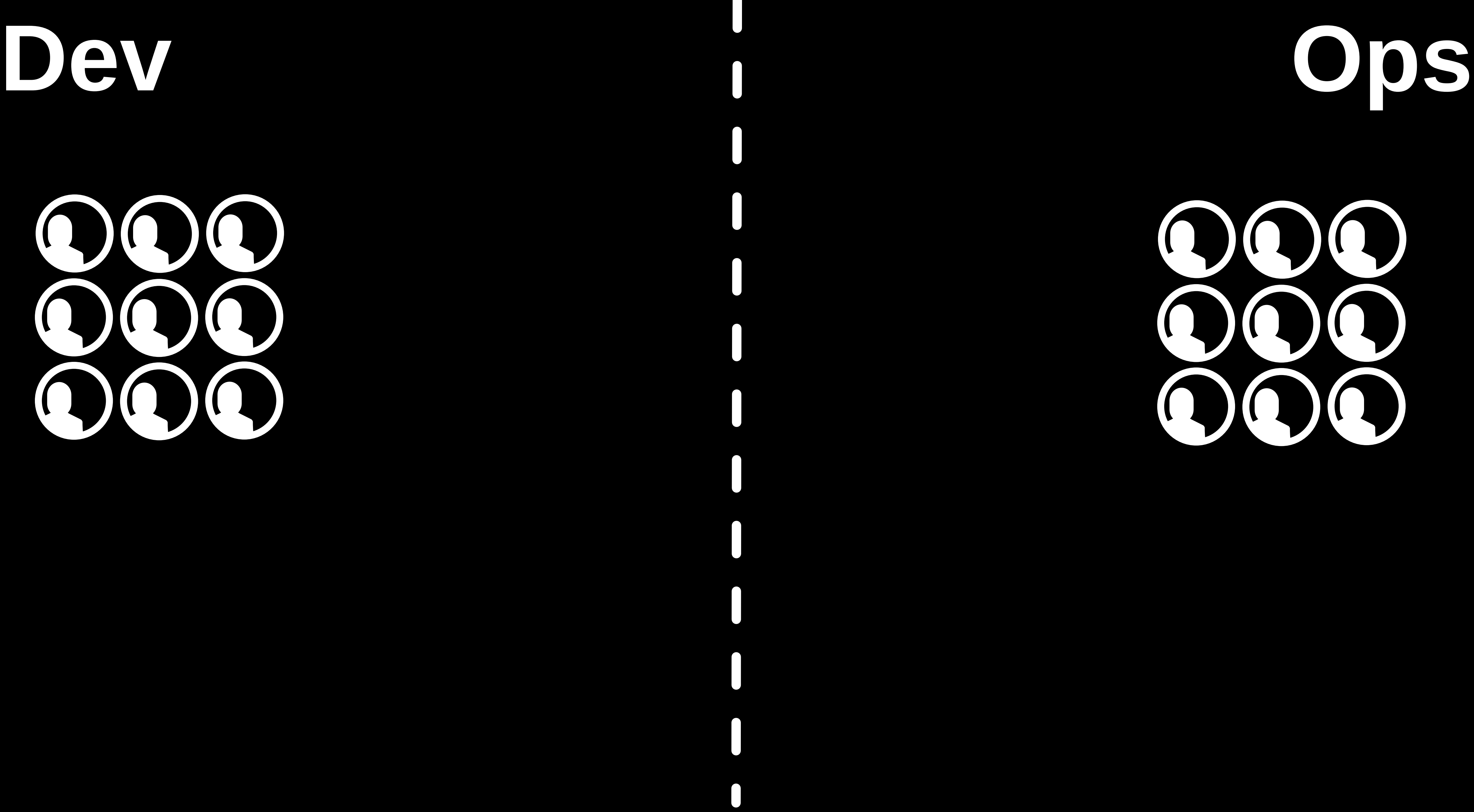
Ops

A culture and mindset for collaboration

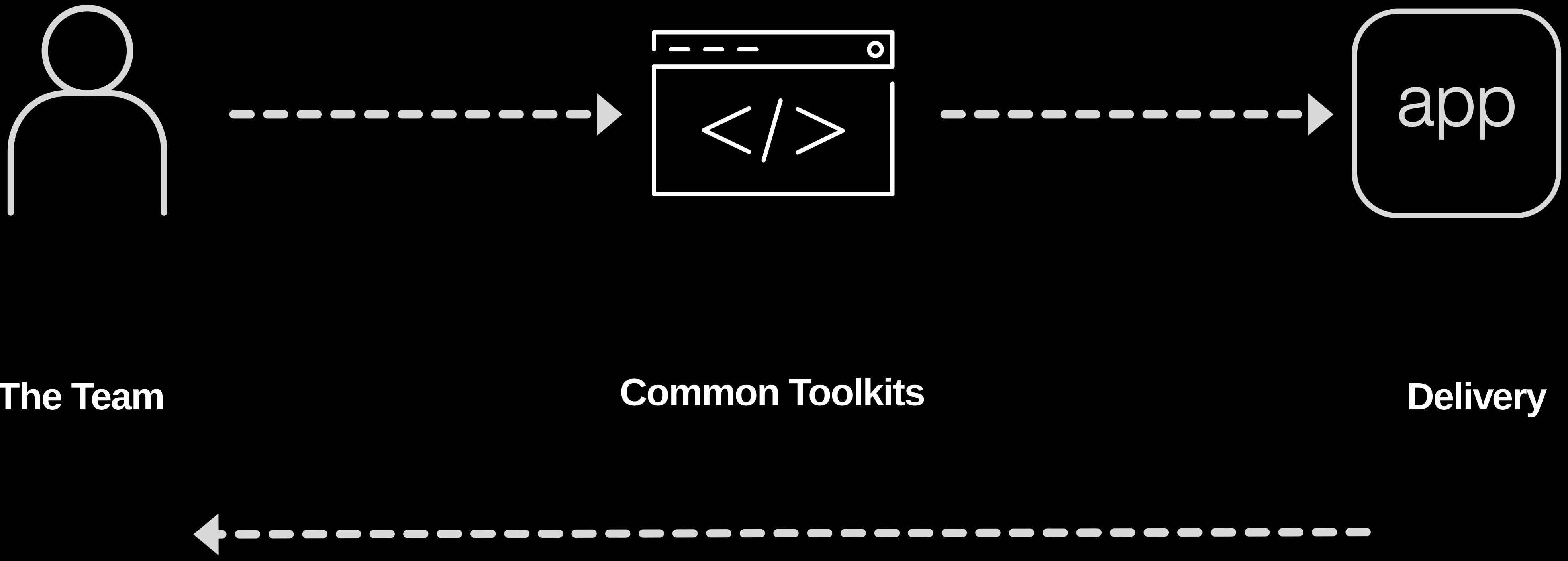
Communication through relationships

Understanding and participating together throughout the entire service lifecycle

A common, integrated process


Dev
Ops

The Goal – To Deliver Valuable Apps



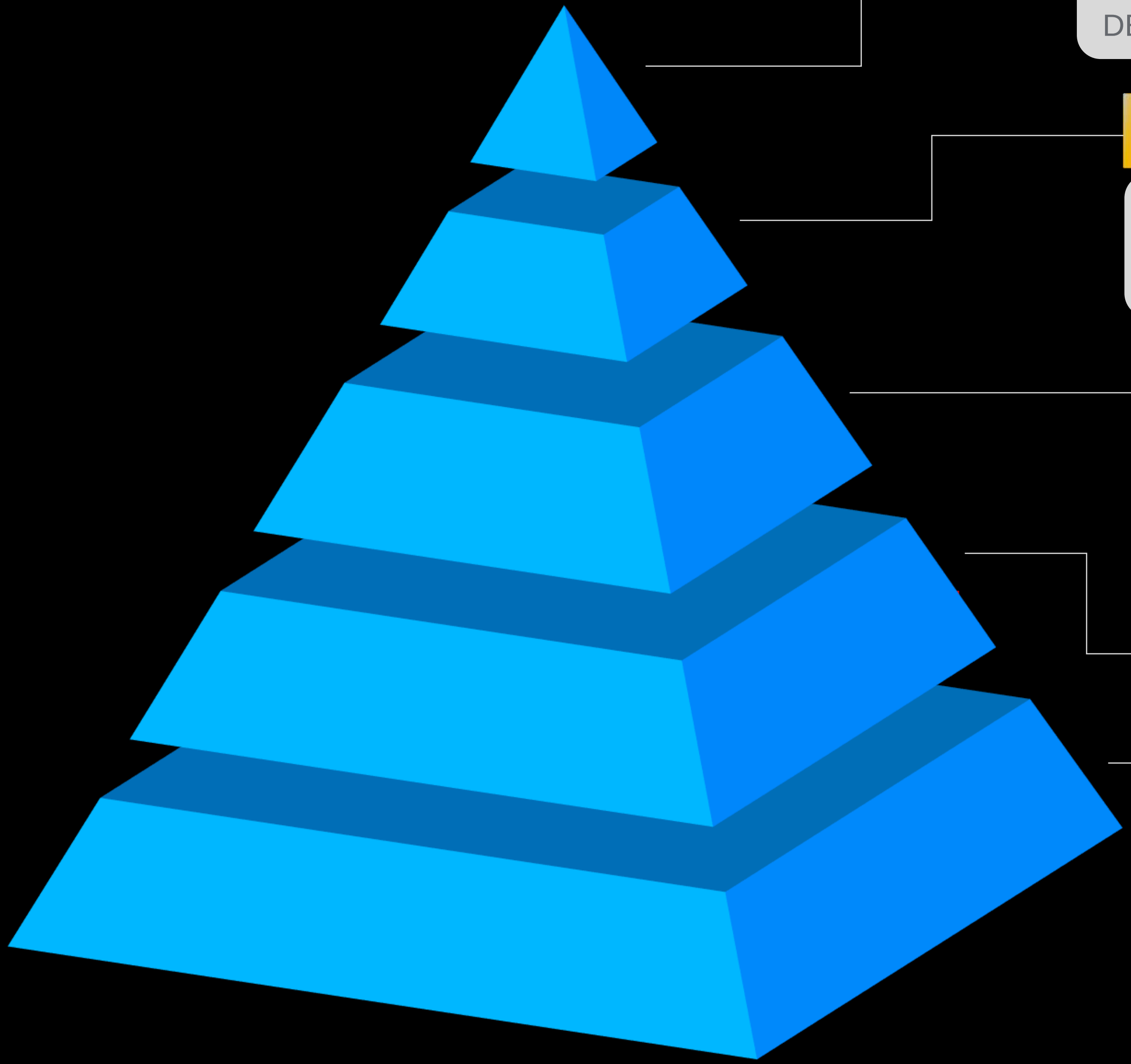
The Team

Common Toolkits

Delivery

© 2017 F5 Networks

Automation
Layers of Abstraction



Jenkins

Self-Service or CI/CD, elevated testing and validation above Code.
DECLARATIVE

Ansible

Toolchain/Toolkit/Pipeline is introduced. Fundemental shift to Code.
DECLARATIVE

BIG IQ

Technology bridge for 3rd parties, acting as a perimeter for our solution. Service/Tenant
DECLARATIVE

iApps

Templates of configrations are developed, standardization of process begins.
IMPERATIVE

BIG-IP

Role Based Access Control Needed, deep knowledge required.
IMPERATIVE

WE MAKE APPS

GO

FASTER. SMARTER. SAFER.

SOLUTIONS FOR AN APPLICATION WORLD