QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2017

灵雀云持续集成和持续交付功能实践

Daniel Morinigo

主办方 Geekbang. 极客邦科技 InfoQ

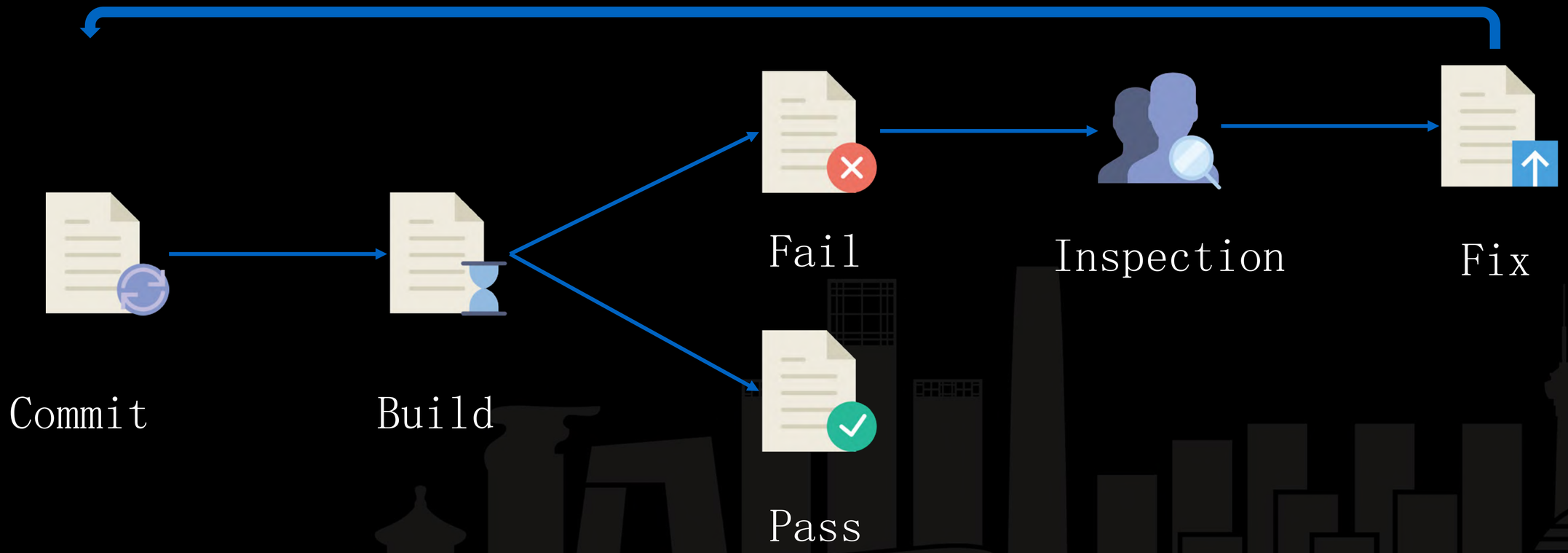
alauda
灵雀云 · 容器服务

DevOps
Continuous Integration
Continuous Delivery

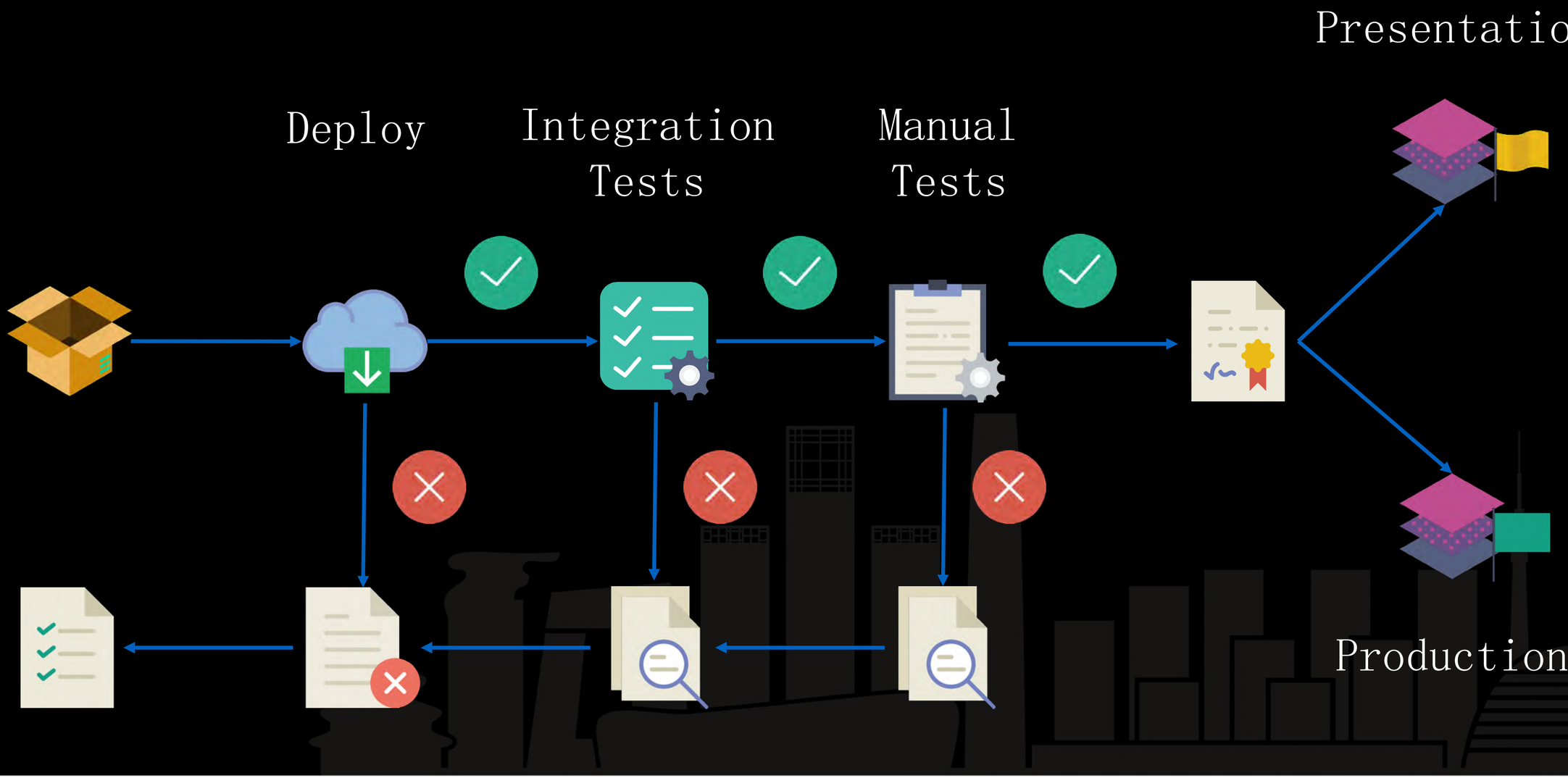
What is DevOps?

- DevOps team:

Continuous Integration



Continuous Delivery



alauda
灵雀云 · 容器服务

Production Quality Images

alauda
灵雀云 · 容器服务

Quick Start

“I never used Docker before, how to get started?”

Lookup for “onbuild” images.
It will easily get you started on building Docker images using only the source code.

Your first Dockerfile could have one line:

12 lines (8 sloc) | 233 Bytes

```
1  FROM golang:1.8
2
3  RUN mkdir -p /go/src/app
4  WORKDIR /go/src/app
5
6  # this will ideally be built by the ONBUILD below ;)
7  CMD ["go-wrapper", "run"]
8
9  ONBUILD COPY . /go/src/app
10 ONBUILD RUN go-wrapper download
11 ONBUILD RUN go-wrapper install
```

Dockerfile.onbuild

```
1  FROM index.alauda.cn/library/golang:1.8-onbuild
2
```

Quick Start

Building is super easy and can be very helpful to get started or to use during development

But once the image is built, its contents will have:

- All the building tools
- Extra software that might not be used
- Project's code

Besides, the image size is also too big for one simple small project

How to solve this problem?

```
$ docker build -t testgo:onbuild -f Dockerfile.onbuild .
Sending build context to Docker daemon 3.072 kB
Step 1/1 : FROM index.alauda.cn/library/golang:1.8-onbuild
# Executing 3 build triggers...
Step 1/1 : COPY . /go/src/app
Step 1/1 : RUN go-wrapper download
 ---> Running in b7da3d260046
+ exec go get -v -d
Step 1/1 : RUN go-wrapper install
 ---> Running in 550cc85209fe
+ exec go install -v
app
 ---> 5e4514126b97
Removing intermediate container 1cb1bcffeb4c
Removing intermediate container b7da3d260046
Removing intermediate container 550cc85209fe
Successfully built 5e4514126b97
```

REPOSITORY	TAG	SIZE
testgo	onbuild	704 MB

alauda
灵雀云 · 容器服务

Builder Pattern

- Two separate Dockerfiles
- **Dockerfile.build** for compiling source
  - Cache dependencies
  - Install building tools or SDKs

- **Dockerfile** for production runtime:
  - Only necessary software

- Using Alauda CI to build the project

- Lightweight images in production:
  - Faster to push/pull
  - Faster to deploy

Dockerfile

```
FROM alpine:3.4

COPY app /app

RUN chmod +x /app

CMD ["/app"]

```

alaudaci.yml

```
version: "0.1.0"
pre_ci_boot:
    image: index.alauda.cn/library/golang
    tag: 1.8-alpine
ci:
    - cp -r . $GOPATH/src/app && cd $GOPATH/src/app
    - go build -ldflags "-w -s" -v -o $ALAUDACI_DEST_DIR/app
    - cp Dockerfile $ALAUDACI_DEST_DIR/Dockerfile

```

alauda
灵雀云 · 容器服务

Dockerfile optimization

- **Concatenate system commands to avoid extra layers**:
  Instead of:

```
RUN apt-get update
RUN apt-get install curl -y
```

  Use:

```
RUN apt-get update && apt-get install curl -y
```

- **Cleanup what you don’t need installed**:
  Libraries and software used during build time, but not on run time
- Install any necessary debugging tools
- **Never** keep secrets inside the image

alauda
灵雀云 · 容器服务

Dockerfile optimization

- **Shared software among projects:**
  Create a base image:

```
FROM debian:jessie
RUN apt-get install curl nginx python mysql-driver -y
```

  In your projects:

```
FROM index.alauda.cn/alauda/base:latest
```

- **These changes will:**
  - Speed up building and deployment processes;
  - Reduce the amount of storage used on a Private Registry;
  - Simplify development and deployment;

Alauda's Continuous Delivery

alauda
灵雀云 · 容器服务

History

- At Alauda we experienced many different ways to deploy software setting different goals:
  - Multiple hosts in a cluster – High availability;
  - Zero-down time - Updates should be smooth;
  - Reliable – Should work;
  - Easy – Anyone can deploy changes;

- We tried many different approaches:
  - Scripting;
  - Hacks;
  - Ansible;

History

- At the beginning it worked, but, with several drawbacks:
- Deploy scripts constantly changing. A typo in the wrong place and it could easily break our production environment; So we also had to **Test the solution to deploy software**;
- Solutions like Ansible are just too brute:
  - Remove old version -> Deploy new version (possible down-time);
- Steep learning curve for beginners:
  - Constant deployment stress;
- Not compatible with Microservices architecture;

alauda
灵雀云 · 容器服务

History

Monolithic Architecture	Monolith Container	Microservices	Microservices	Microservices
Ansible	Ansible + Docker	Ansible + Docker	Alauda	Alauda CI/CD
Manual testing	Manual testing	Manual testing	Automation + Manual testing	Unit tests + Automation + Manual testing
2x week 2 hours	1~2x week 1 hour	5x week 1 hour	10x week 30 minutes	20x week 10 minutes

Alauda's Pipeline

Design

- Containerized:
  - A traditional pipeline will deal with **Artifacts**, ready to deploy software;
  - In a containerized environment Docker Images are the **artifacts**;

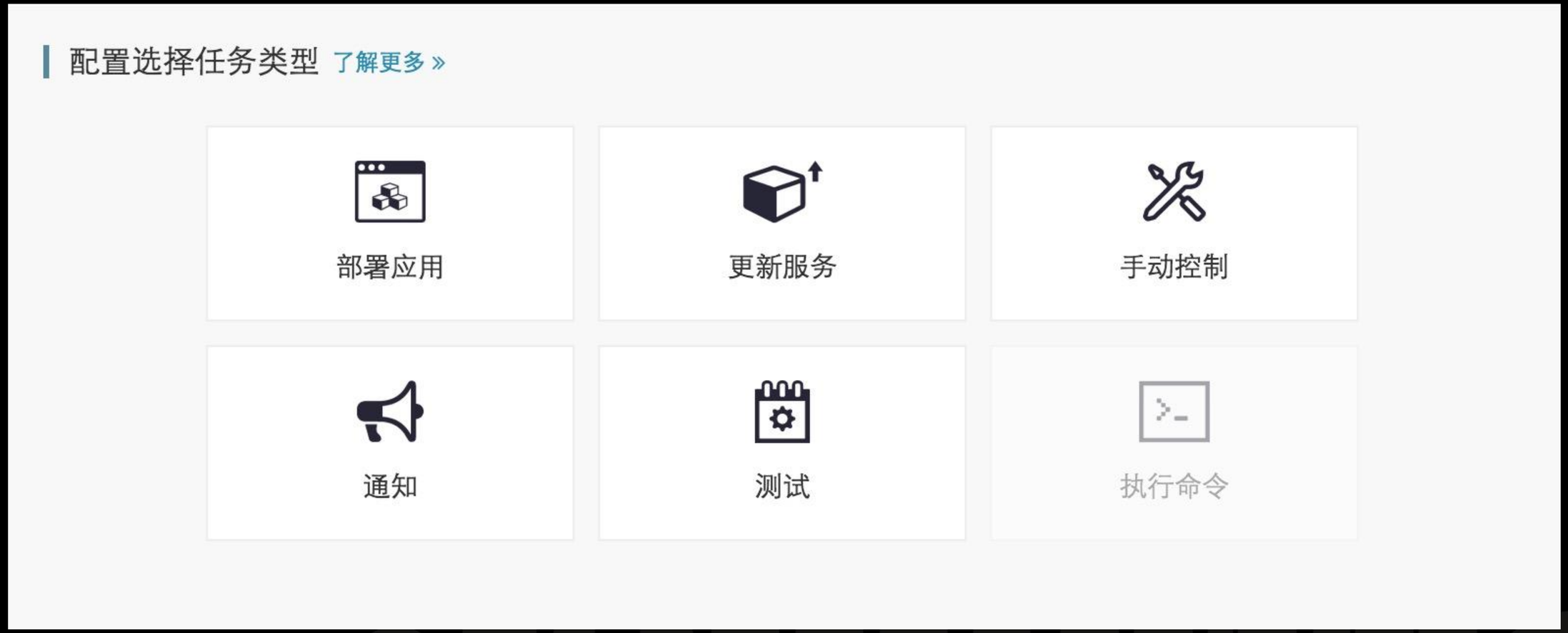
- Atomic:
  - A pipeline is composed by different tasks;
  - Multiple task types;
  - Up to you on how to compose those tasks;

- Easily satisfy different needs and scenarios;

- From Development to Production;

alauda
灵雀云 · 容器服务

Task Types

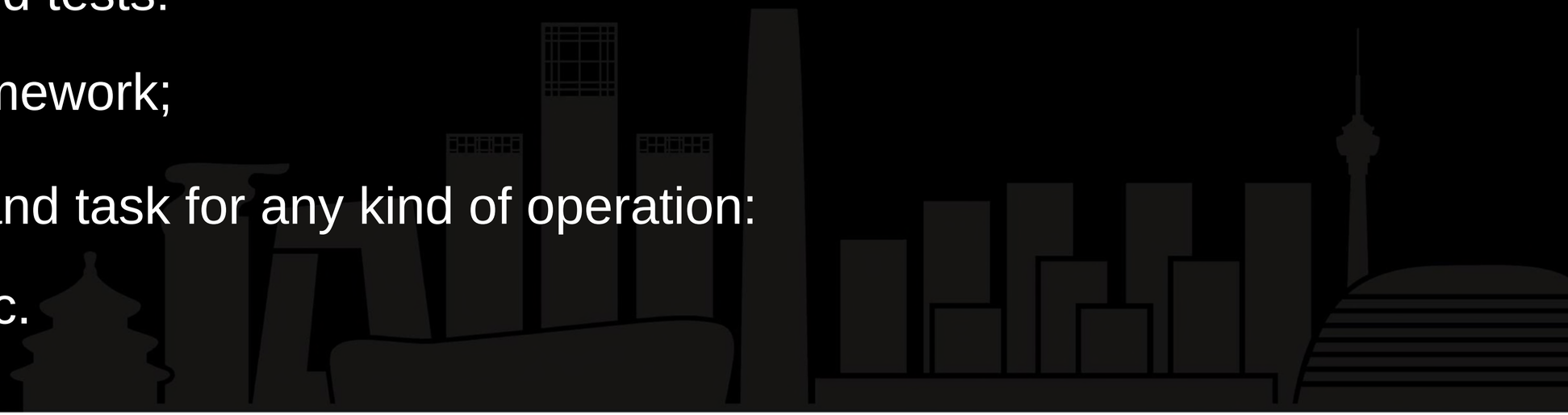


Deploy Application

- Deploys a temporary application using a template and updating the image of the selected services;
- Use-cases:
- Run Integration tests against a full application stack:
  - Microservices architecture with multiple services;
  - Test support using different types of backend storages;
  - Run business focused tests over a controlled data set;
  - Run calculations over a specific application structure;

Automated Tests

- Deploys a one-time task container on a cluster;
- Use-cases:
- Run automated test suites against a service/application:
  - Integration tests;
  - End-to-end tests;
  - UI automated tests:
    - Selenium;
    - Robot framework;
- Run on-demand task for any kind of operation:
  - Reports
  - Backups, etc.



alauda
灵雀云 · 容器服务

Update Service

- Updates an already running service;
- Keep consistent and running services with automatic rollback;
- Automatically reflect changes of a environment variables file to a service during update;
- Easily update a test/stage/production environment to speed-up delivery speed;

Manual Control

- Interrupts a pipeline for a limited amount of time and requires manual approval or interruption;

- Use-cases:
  - Manual tests;
  - Approval for release;
  - Control time-sensitive releases;
  - Time-buffer for assisted deployment;

Notifications

- Sends a notification with pipeline related data;

- Notifications support:
  - Email;
  - SMS messages;
  - Webhooks;

- Use-cases:
  - Notify team members regarding a specific phase of release;
  - Automate remote command execution using webhooks;
  - Alert oncall to assist deploy;

alauda
灵雀云 · 容器服务

Case: Some Alliance

Challenges:

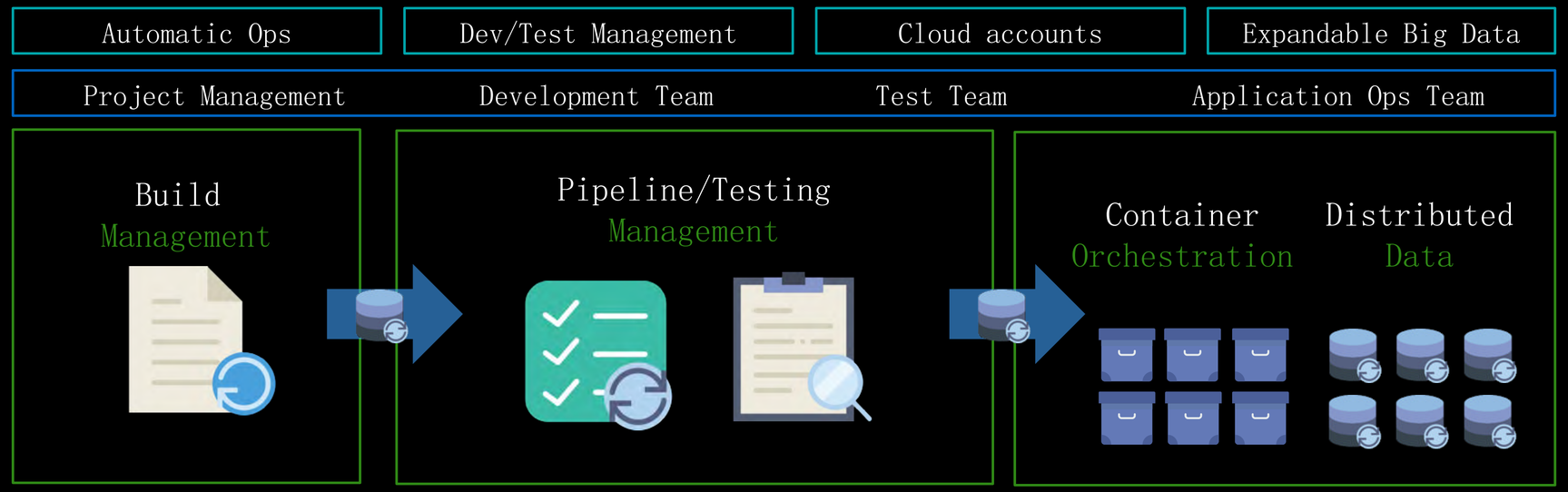
- Product's life very complex:
  - Manual packaging and deployment of software;
  - Differences among environments leading to failure on Production;
  - Too much time spent on deployment;

- Traditional IT system:
  - Complex to deploy Testing environment;
  - Complex maintenance of test data;
  - Management of third-party software's sandbox;

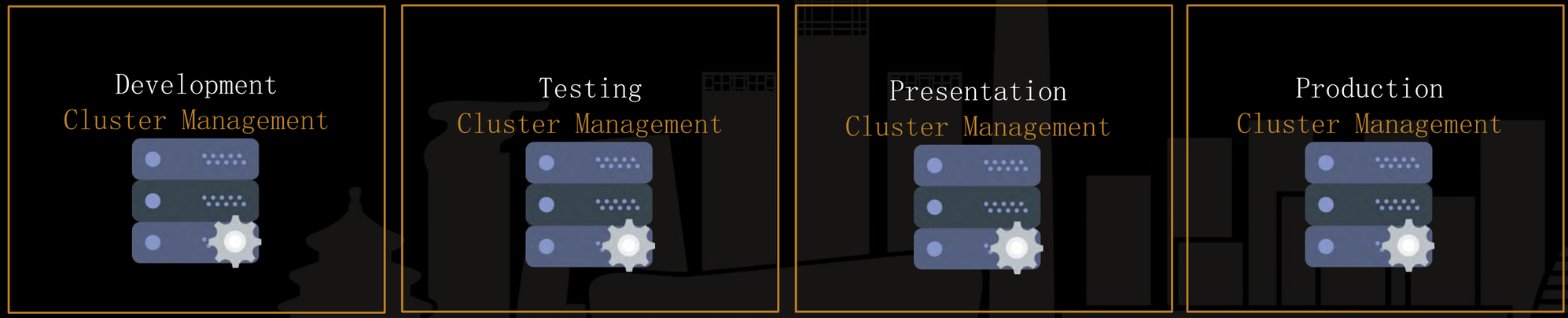
- Operational support for alliance's members' system:
  - Responsible for development, maintenance and operation of multiple systems;
  - Each system's deployment is unique;
  - Multiple services per system
  - System's service stack is not uniform;

alauda
灵雀云 · 容器服务

Case: Some Alliance



Infrastructure management



alauda
灵雀云 · 容器服务

Case: Some Alliance

Solution:

- Use Alauda's Pipeline:
  - Define system's services and third-party software deployment process;
  - Automate deployment of software to the Test, Staging, and Production environments unifying and setting the software lifespan;
  - Use application templates to unify services;
  - Use different volumes to manage test data;
  - Manage different cluster for each associated companies sharing the same images;

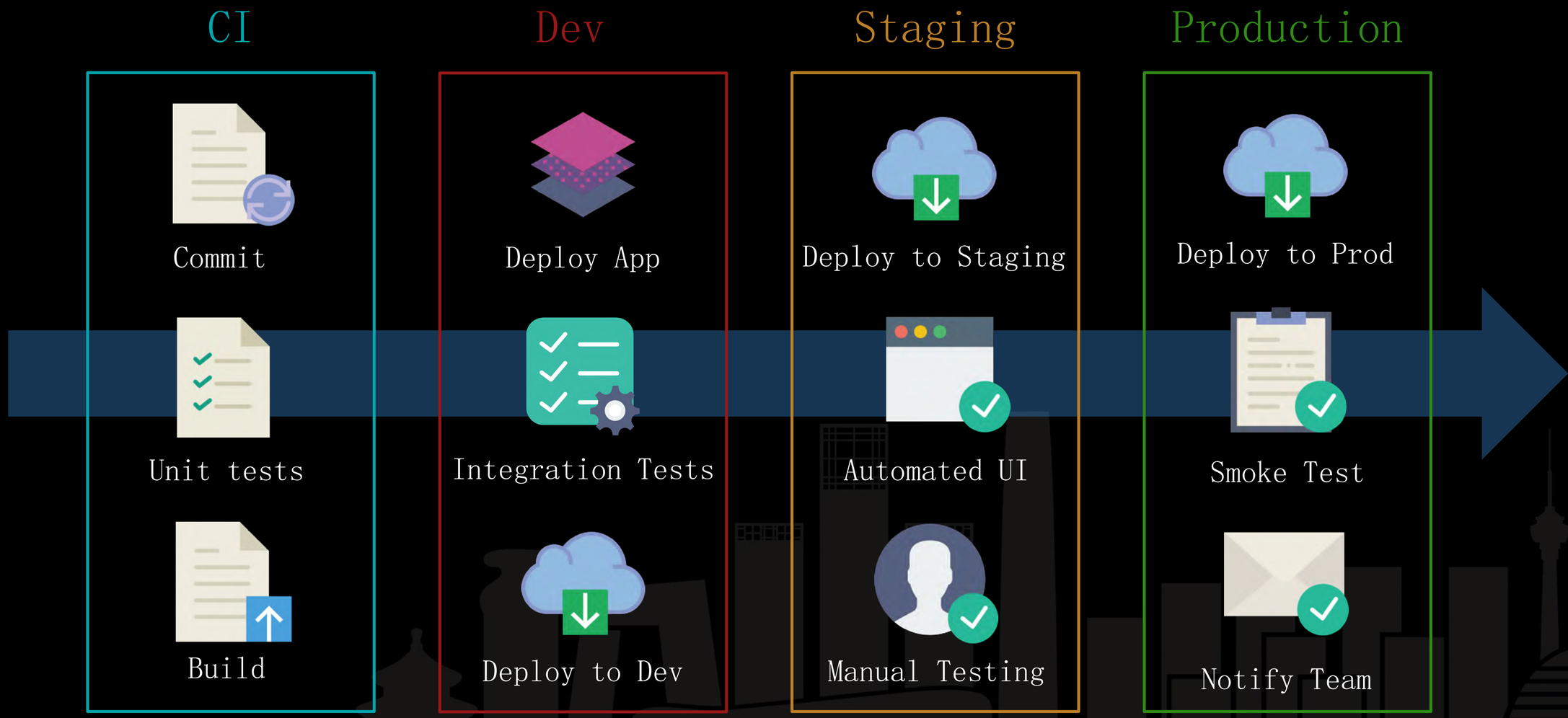
Customer Benefits:

- Standardize software deployment process:
  - Automate deployment, rollback a tracing of software deployment and versions;
  - Improved software publishing quality;

- Fast Test environment deploys:
  - Test environments can be requested at any time by the associated companies or vendors;
  - Test versions are specified and necessary test data is deployed automatically;
  - Reduced immensely software testing cycles;

- Unified cluster management solution:
  - Speedup monitoring, alarm and authorization for all different associated companies;
  - Easily maintain and keep track of application version;

Pipeline


CI
Dev
Staging
Production
Commit
Unit tests
Build
Deploy App
Integration Tests
Deploy to Dev
Deploy to Staging
Automated UI
Manual Testing
Deploy to Prod
Smoke Test
Notify Team

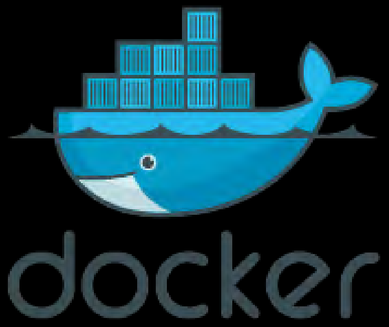
Metrics-driven Development

- Continuous Tests:
  - Continuously mimics user behavior on all environments of the pipeline
  - Generates metrics that are constantly monitored
  - Test team like dev/ops:
    - Write test code and update all environments;
    - Monitor the quality of new versions through generated metrics;
  - Product instrumented with all sorts of metrics:
    - Triggered by user behavior tests
    - Alerts triggered while the release is deployed throughout our clusters
    - Easily detect bugs through metrics

Alauda CI/CD

Private Repository

Continuous Integration



Private Registry

Continuous Delivery

Questions?

关注QCon微信公众号，
获得更多干货！

Thanks!

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

主办方 Geekbang. 极客邦科技 InfoQ