Golang, Micro Service, Continuous Delivery and Docker

郭峰
联合创始人@DaoCloud

DAOCLOUD

README.md

Development Challenge

Solution of Challenge: **Micro Service**

Challenge of Solution of Challenge

Solution of Challenge of Solution: **Docker**

Demo

DAOCLOUD

Iteration 1

```go
func main() {
    http.HandleFunc("/", Hello)

    fmt.Println("Start listening...")

    err := http.ListenAndServe(":4040", nil)
    if err != nil {
        panic(err)
    }
}

func Hello(res http.ResponseWriter, req *http.Request) {
    fmt.Fprintln(res, getHelloMsg())
}

func getHelloMsg() string {
    return "Let's Shopping!"
}
```

Iteration 1

```go
func TestGetHelloMsg(t *testing.T) {
    msg := getHelloMsg()
    if msg != "Let's Shopping!" {
        t.Error("Get hello message error")
    }
}
```

> go install github.com/DaoCloud/shop

> go test github.com/DaoCloud/shop

CHEERS

Iteration 2

```go
func GetPerson() string {
    conn := "admin:admin@daocloud.io"

    session, err := mgo.Dial(conn)
    if err != nil {
        panic(err)
    }
    defer session.Close()
    session.SetMode(mgo.Monotonic, true)

    // Collection People
    c := session.DB(db).C("people")

    result := Person{}
    err = c.Find(bson.M{"name": "DaoCloud"}).Select(bson.M{"phone": 0}).One(&result)
    if err != nil {
        panic(err)
    }
    return fmt.Sprintf("%v", reflect.ValueOf(&result).Elem().Field(1))
}
```

Iteration 2

```
var _ = Describe("Shop", func() {
    Describe("Adding and retreving Person object from MongoDB", func() {
        Context("inspecting their name", func() {
            It("should result 'DaoCloud'", func() {
                Expect(GetResult()).To(Equal("DaoCloud"))
            })
        })
    })
})
```

```
> go install github.com/DaoCloud/shop
```

```
> go test github.com/DaoCloud/shop
```

NOT BAD

Iteration 3

Iteration 4

Iteration 5

Iteration 6

Iteration 7

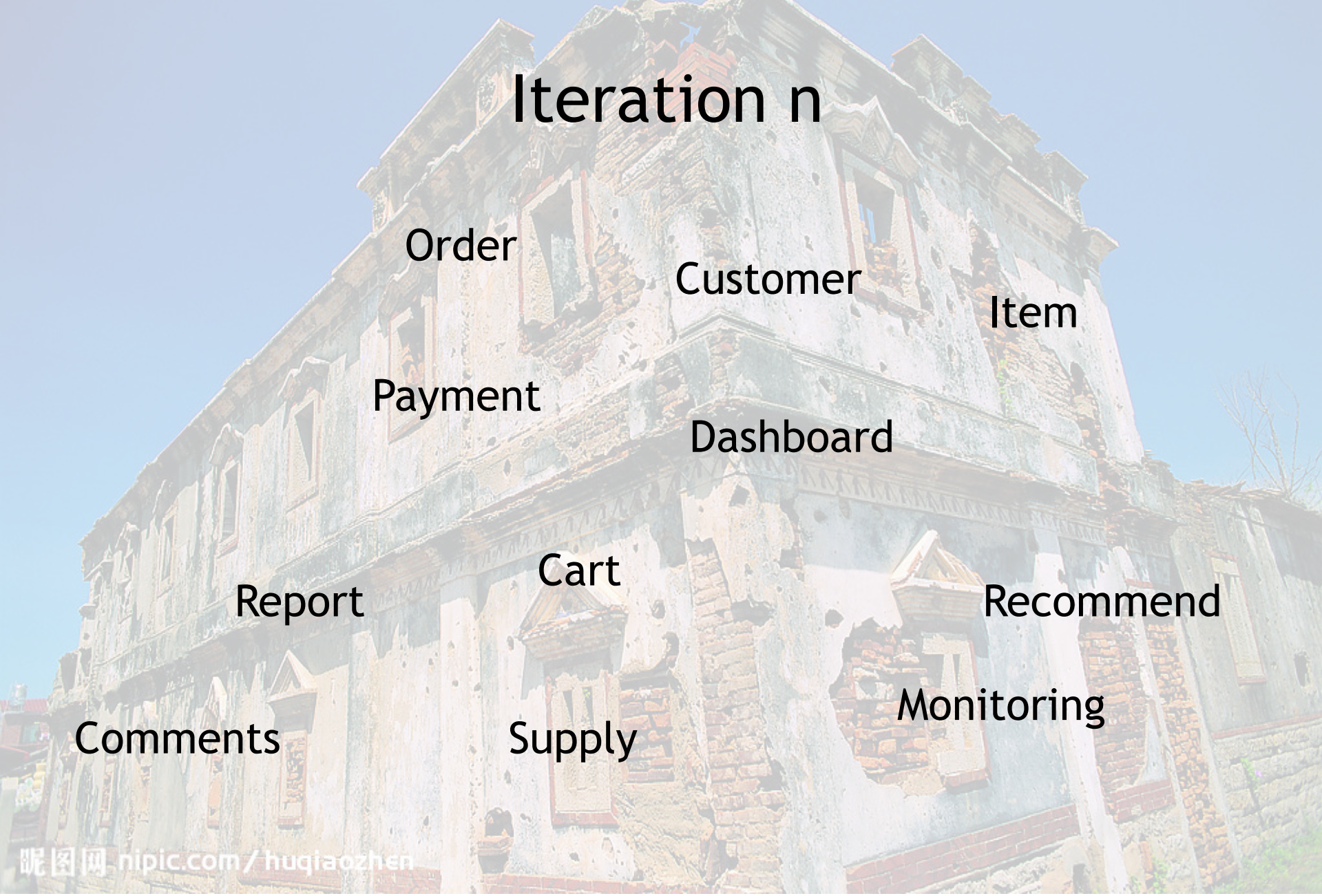
Iteration 8

Iteration 9

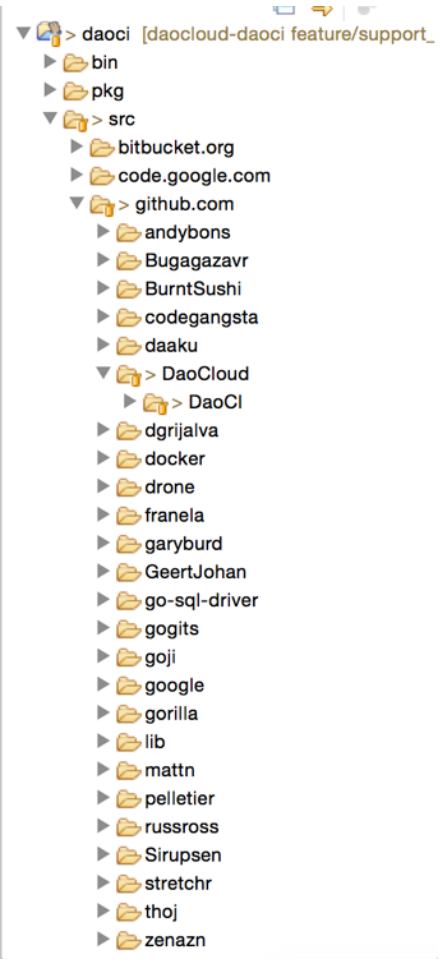
Iteration ...

GET CRAZY

Iteration n


Order
Customer
Item
Payment
Dashboard
Cart
Report
Recommend
Monitoring
Comments
Supply
昵图网 nipic.com/huqiaozhen

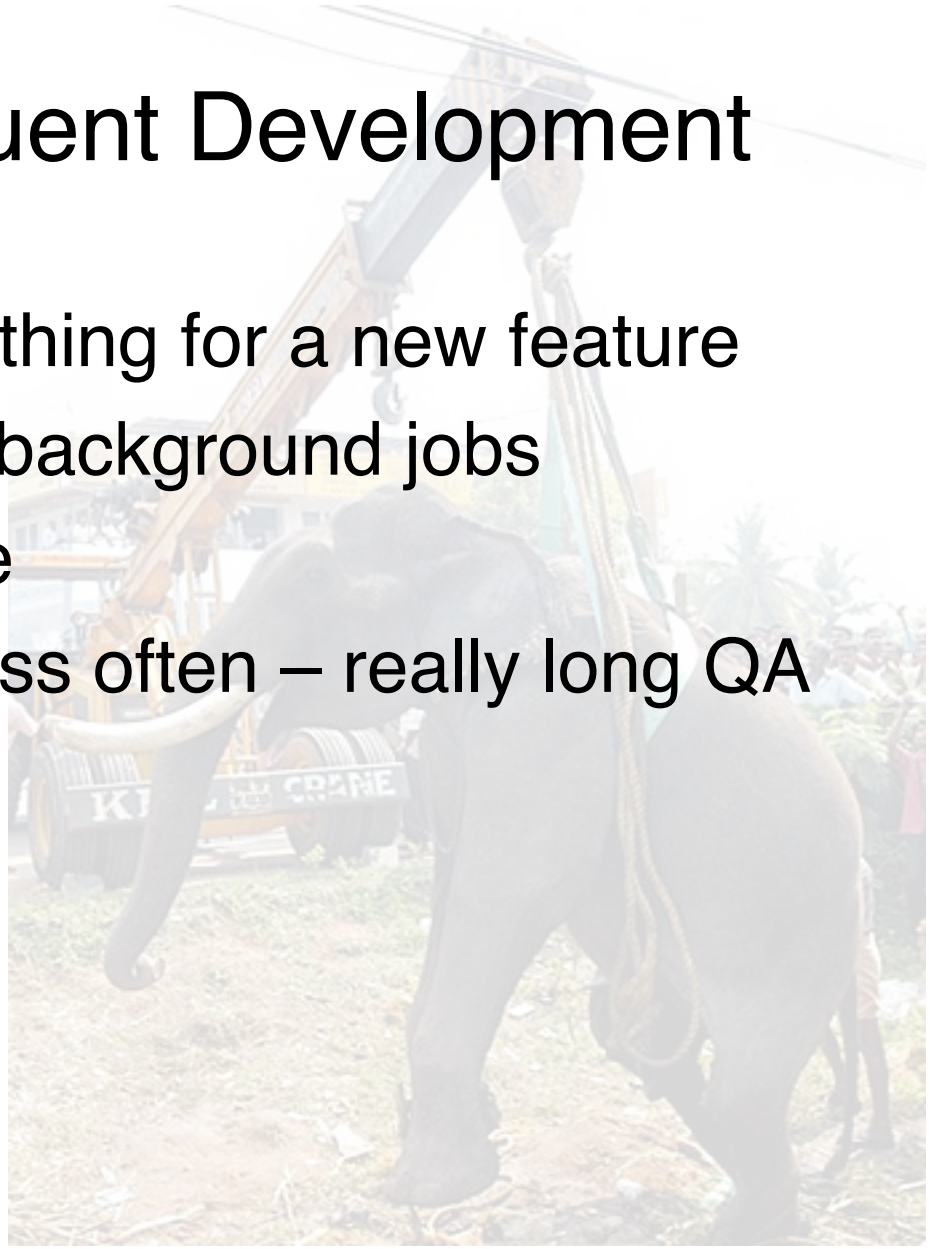
Intimidate Developers



DAOCLOUD

Obstacle to Frequent Development

- Need to redeploy everything for a new feature
- Interrupts long running background jobs
- Increasing risk of failure
- Updates will happen less often – really long QA cycles

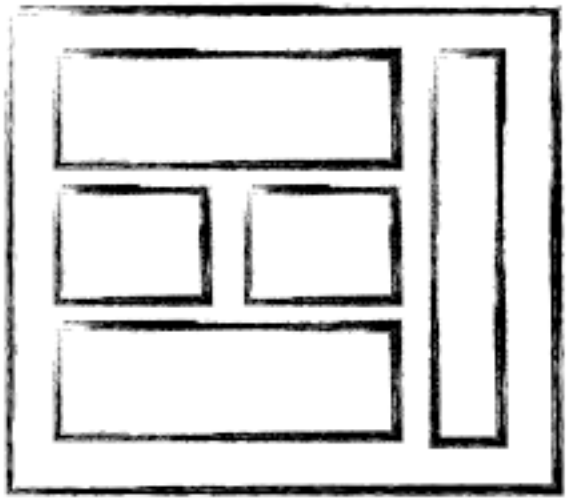


DAOCLOUD

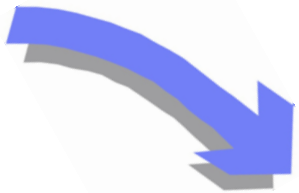
Overloads Your IDE and Focus

DAOCLOUD

Require long-term commitment to a tech stack



Monolithic/Layered



Micro Services

MicroService needs Self-Sufficient

DAOCLOUD

Vendoring/Versioning

http://godoc.org/github.com/tools/godep

```
$ godep save
```

```
{
    "ImportPath": "github.com/kr/hk",
    "GoVersion": "go1.1.2",
    "Deps": [
        {
            "ImportPath": "code.google.com/p/go-netrc/netrc",
            "Rev": "28676070ab99"
        },
        {
            "ImportPath": "github.com/kr/binarydist",
            "Rev": "3380ade90f8b0dfa3e363fd7d7e941fa857d0d13"
        }
    ]
}
```

Import Proxy

gopkg.in
Stable APIs for the Go language

```
gopkg.in/pkg.v3      → github.com/go-pkg/pkg (branch/tag v3, v3.N, or v3.N.M)
gopkg.in/user/pkg.v3 → github.com/user/pkg   (branch/tag v3, v3.N, or v3.N.M)
```

DAOCLOUD

Self-Sufficient ≠ Code-Sufficient

DAOCLOUD

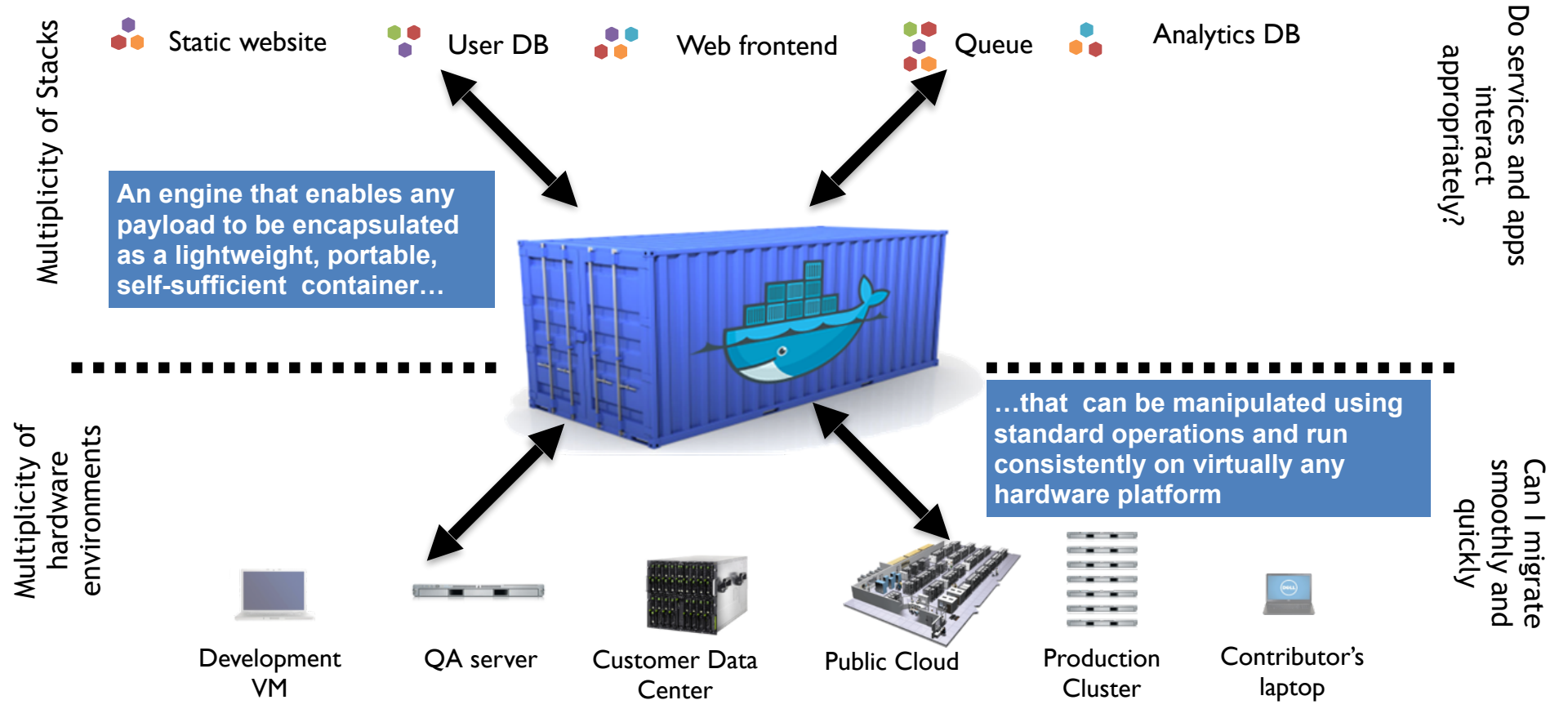
Build, Ship and Run Any App, Anywhere

Docker - An open platform for distributed applications for developers and sysadmins.

* Powered by Linux Containers.

DAOCLOUD

Docker is a shipping container system for code



DAOCLOUD

Self-sufficient in Docker Way

- A Docker container contains everything it needs to run:
  - Minimal base OS
  - Libraries and frameworks
  - Application code

Self-sufficient in Docker Way - Dockerfile

```
# Start from a Debian image with the latest version of Go installed
# and a workspace (GOPATH) configured at /go.
FROM golang:1.3.3

# Copy the local package files to the container's workspace.
ADD . /go/src/github.com/DaoCloud/shop

# Build the app inside the container.
# (You may fetch or manage dependencies here,
# either manually or with a tool like "godep".)
RUN go install github.com/DaoCloud/shop

# Run the app by default when the container starts.
ENTRYPOINT /go/bin/shop

# Document that the service listens on port 4040.
EXPOSE 4040
```

DAOCLOUD

Docker & Micro Service

- ✓ Develop simplest possible solution
- ✓ Configuration is a runtime constraint
- ✓ Not extra-extra-complex application

Dev

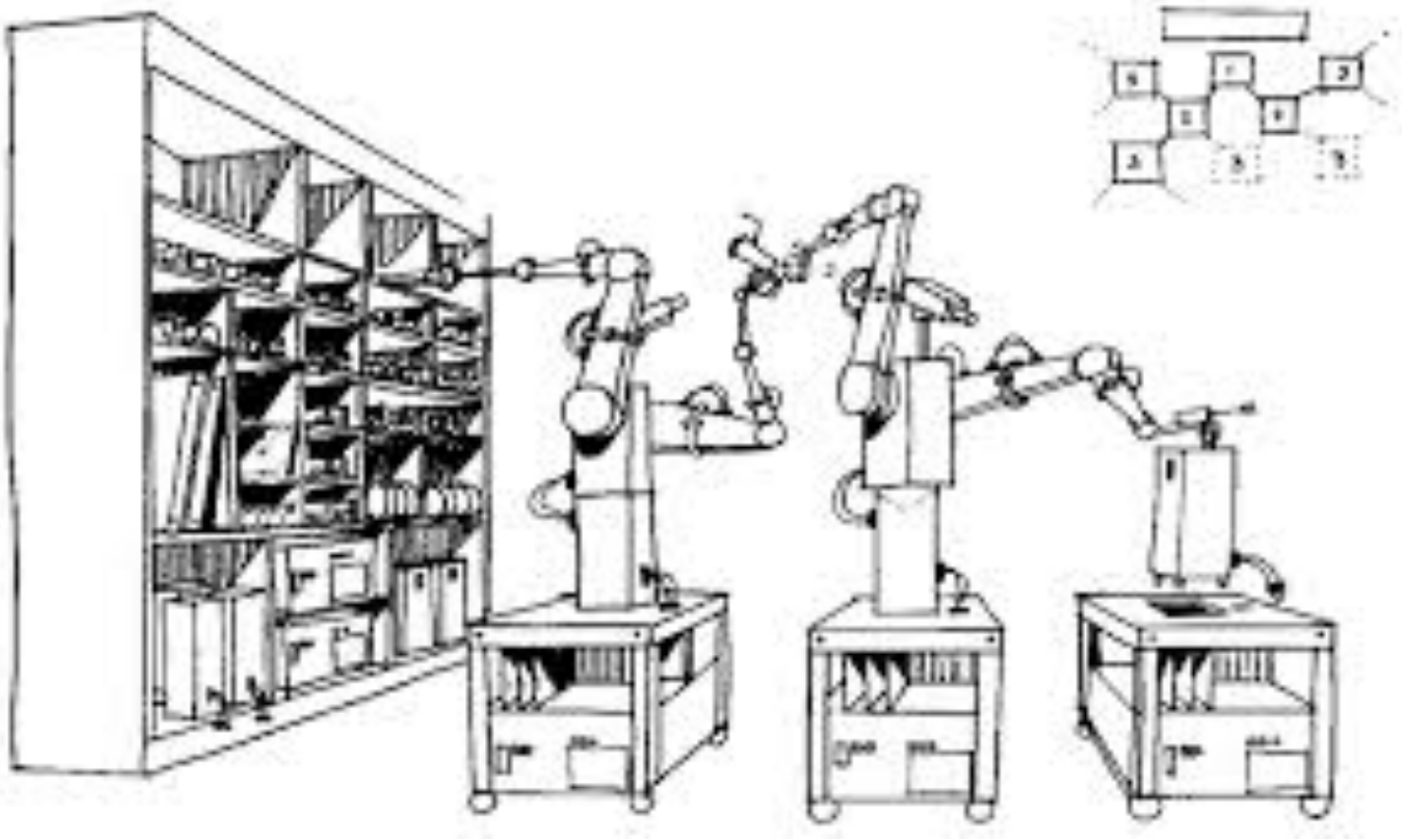
```
new WebServer().start(8080);
```

Ops

- ✓ Manage hardware / infrastructure
- ✓ Monitoring / backups
- ✓ Not apps implementation details

DAOCLOUD

MicroService needs CI/CD

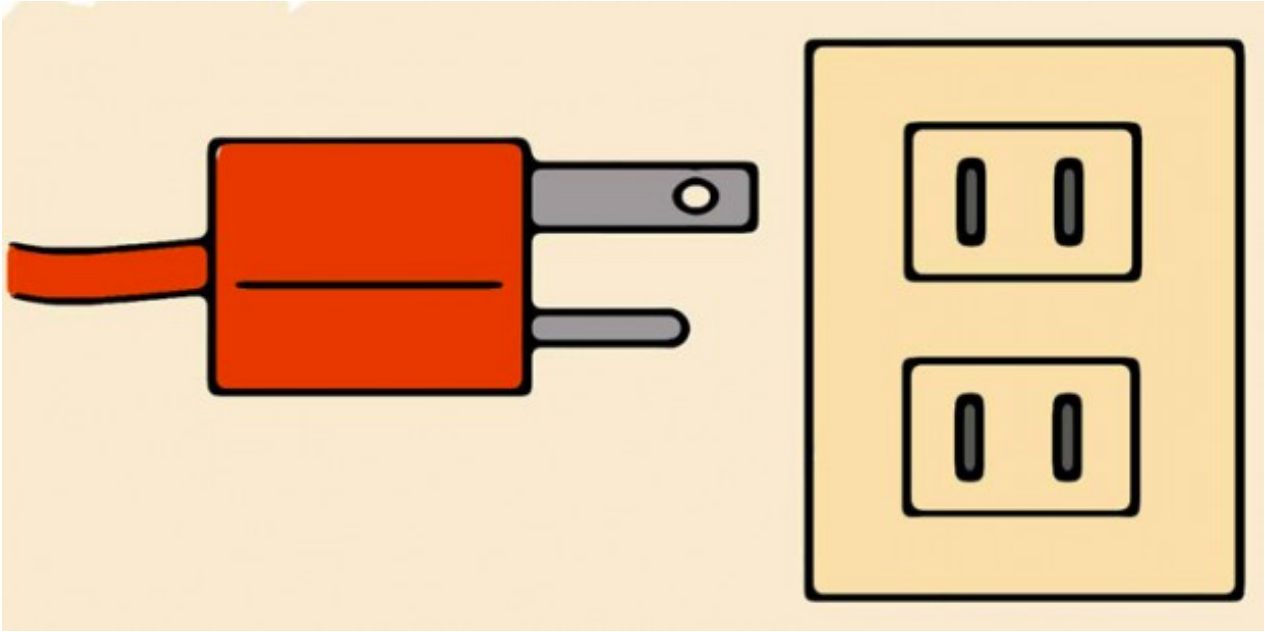


Automation is Implicit

DAOCLOUD

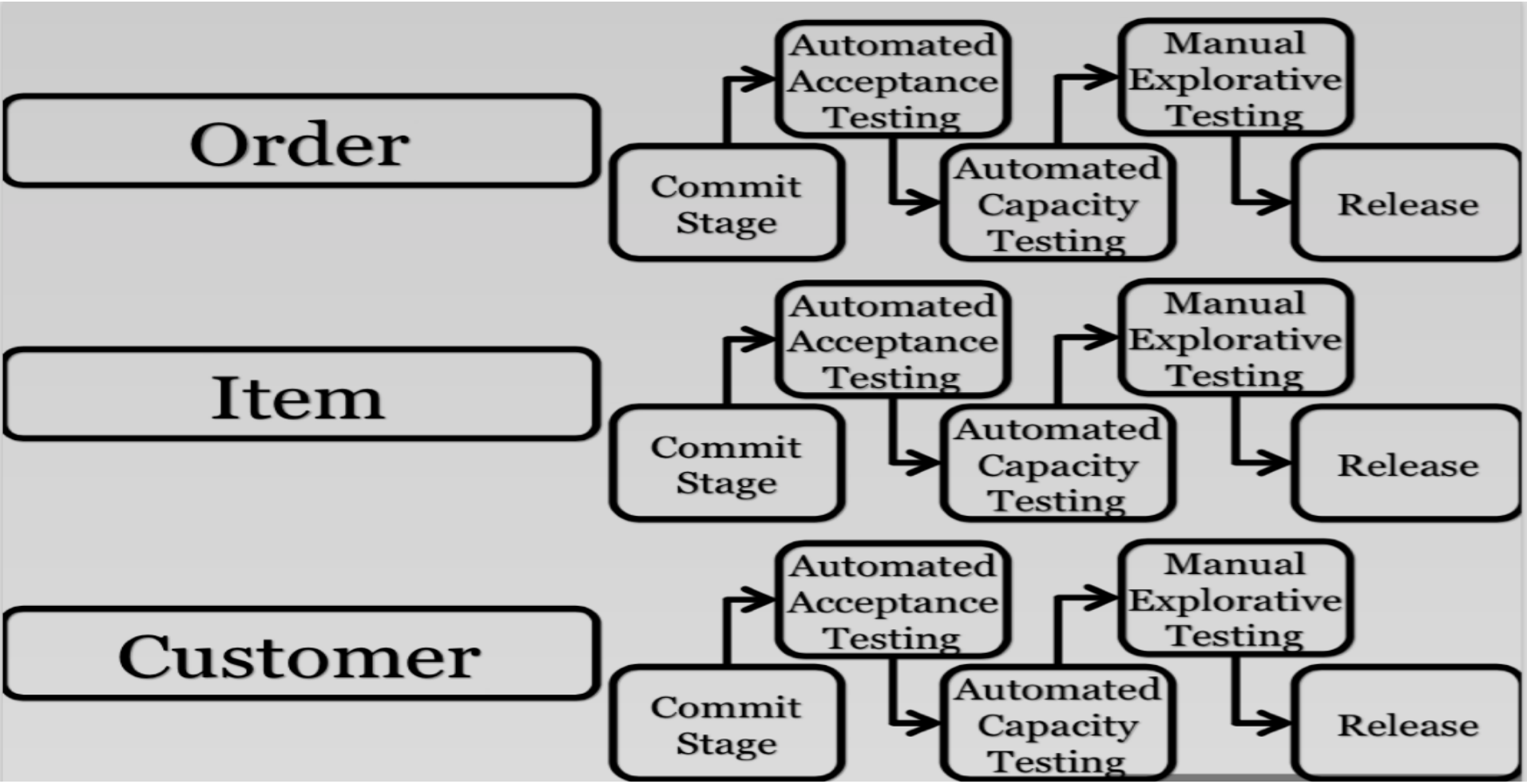
Continuous Integration

Continuous Integration is the practice of integrating early and often, so as to avoid the pitfalls of "Integration Hell".



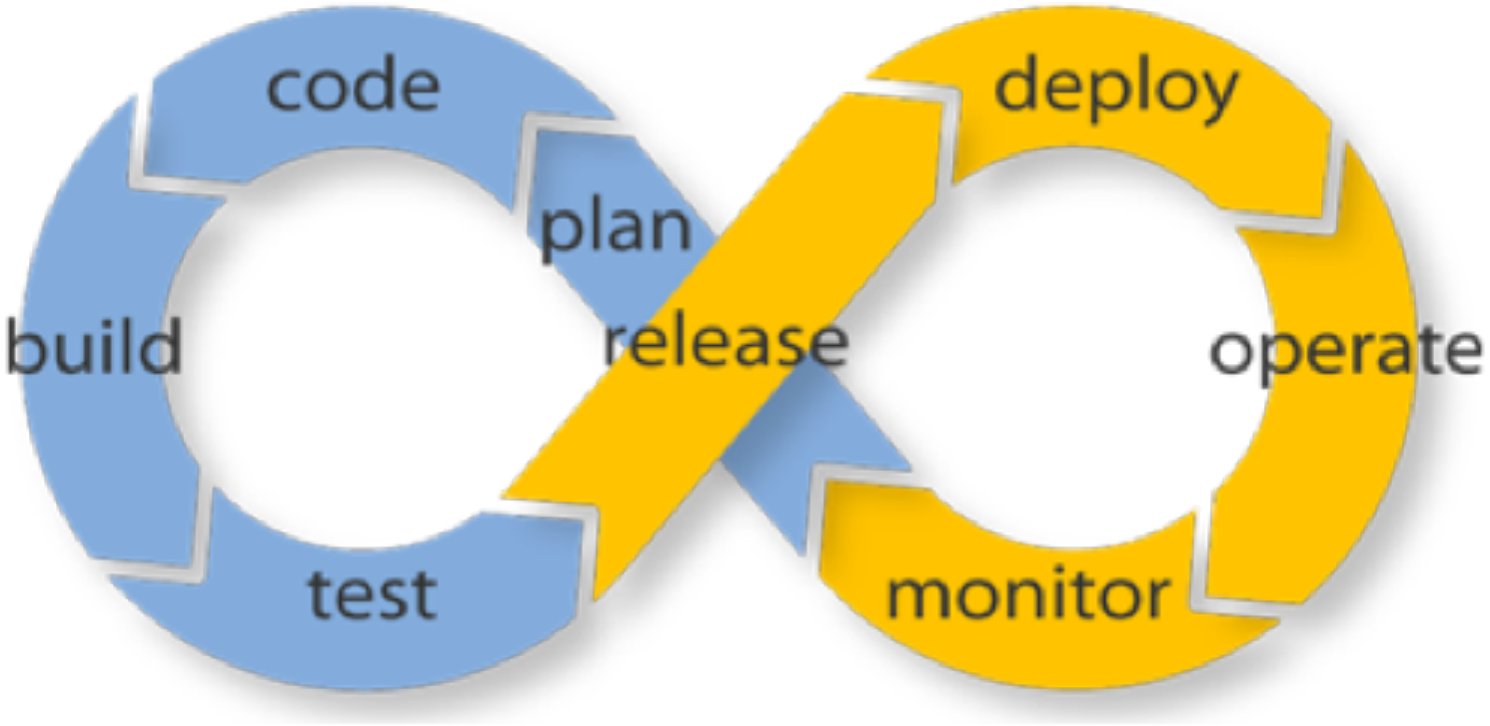
DAOCLOUD

Continuous Integration to Continuous Diverse!

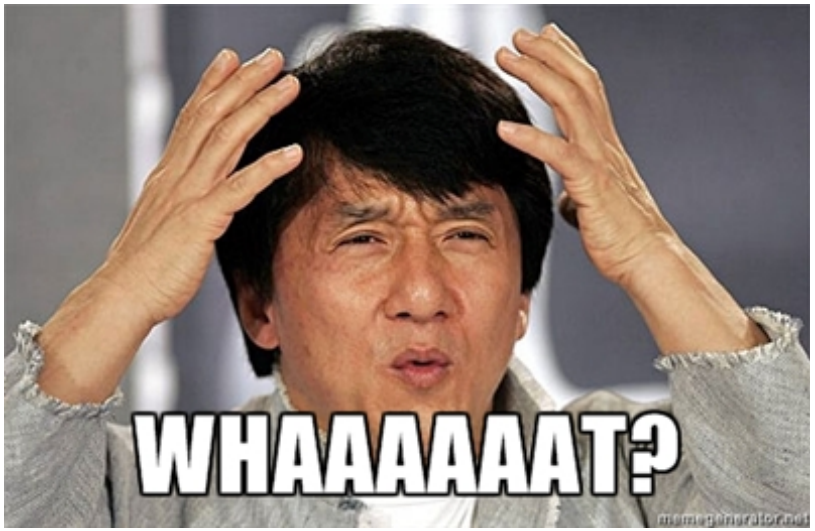


DAOCLOUD

Continuous Integration to Continuous Delivery



Duang, Duang…



“Works for me”
“You need start A first”
“A with version 1.2.5-xx cannot work with B with version 2.0.7”
“So to trigger the bug you have to install X and Y then configure A, B and C, then download the extra file, put it in this directory.

DAOCLOUD

Docker Compose

- One binary to start/manage multiple containers and volumes on a single Docker host
- Originated from Fig
- Move your `docker run` commands to a YAML file

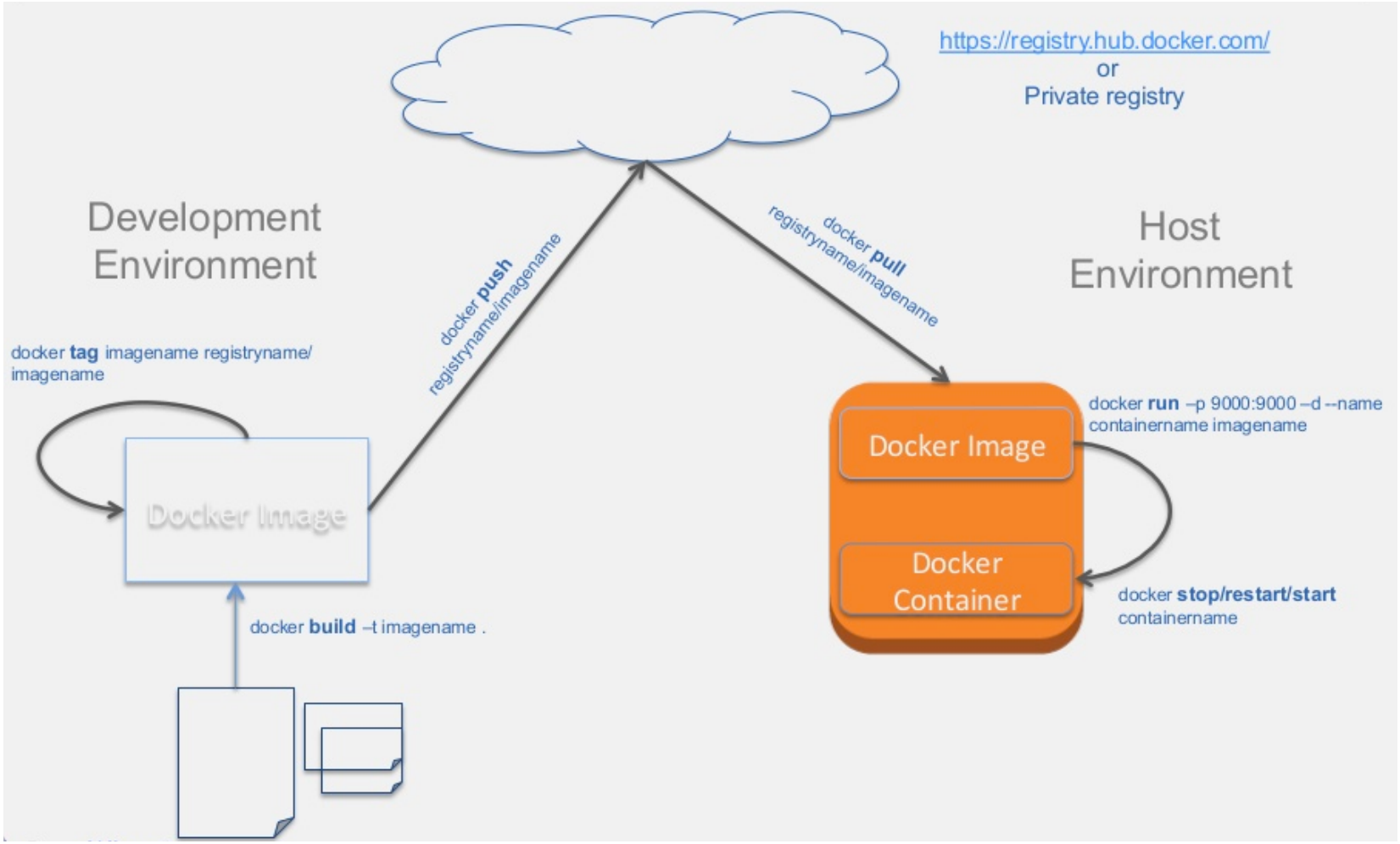
YAML description

```
wordpress:
  image: wordpress
  links:
   - mysql
  ports:
   - "80:80"
  environment:
   - WORDPRESS_DB_NAME=wordpress
   - WORDPRESS_DB_USER=wordpress
   - WORDPRESS_DB_PASSWORD=wordpresspwd
mysql:
  image: mysql
  volumes:
   - /home/docker/mysql:/var/lib/mysql
  environment:
   - MYSQL_ROOT_PASSWORD=wordpressdocker
   - MYSQL_DATABASE=wordpress
   - MYSQL_USER=wordpress
   - MYSQL_PASSWORD=wordpresspwd
```

Use

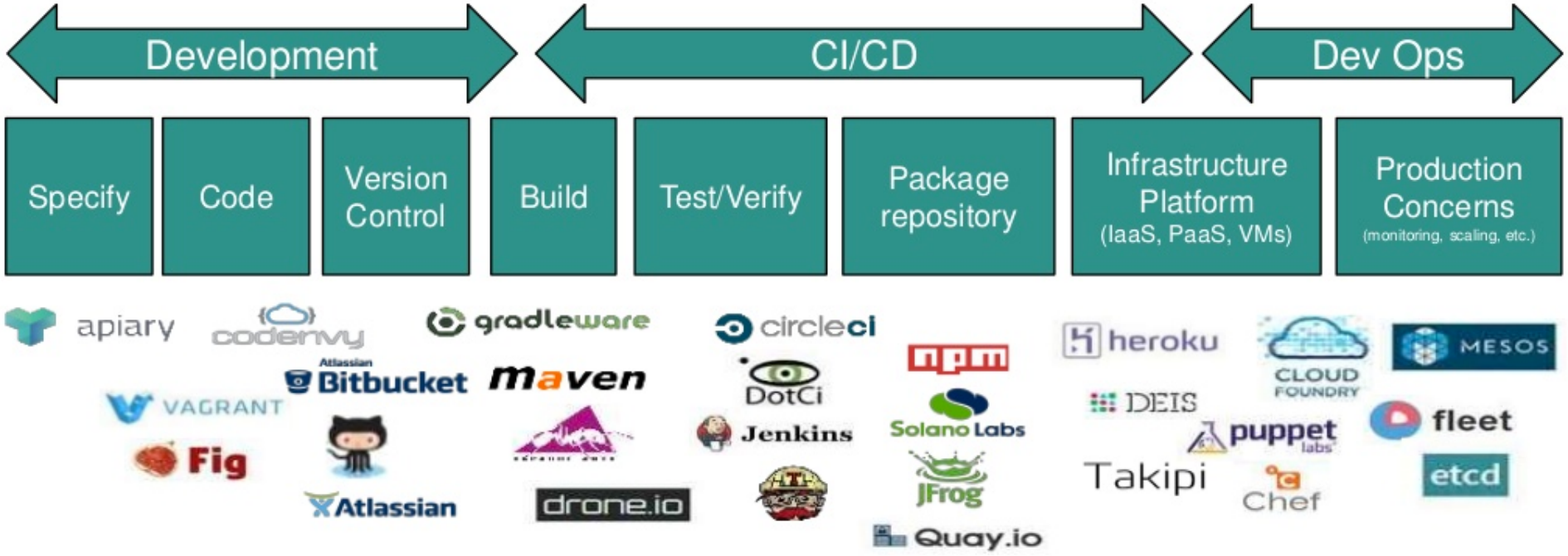
```
$ docker-compose up -d
Creating vagrant_mysql_1...
Creating vagrant_wordpress_1...
$ docker-compose ps
Name                    Command                          State
Ports
-------------------------------------------------------------------
vagrant_mysql_1         /entrypoint.sh mysqld             Up
3306/tcp
vagrant_wordpress_1     /entrypoint.sh apache2-for ...    Up
0.0.0.0:80->80/tcp
```

Reproducible Delivery



DAOCLOUD

The Landscape with Containers



DAOCLOUD

Demo

Q&A

DAOCLOUD