阿里巴巴云原生专场

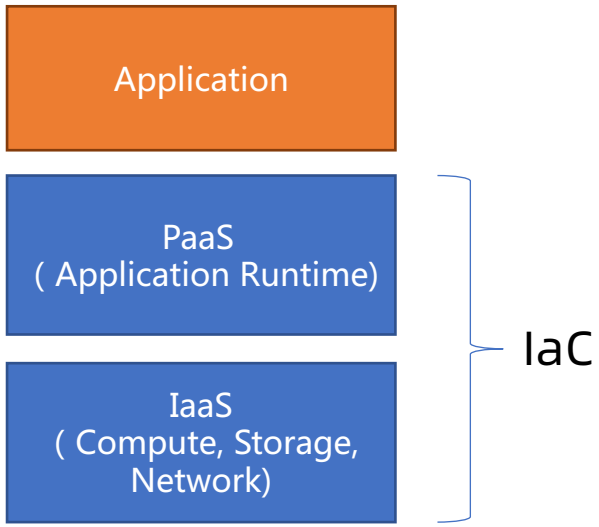
# Infrastructure as Code 在阿里巴巴的初步实践

演讲人：许晓斌

juven.xuxb@alibaba-inc.com

观看视频回放

# Infrastructure as Code（基础设施即代码）




1. 业务的交付，除了业务代码之外，还需要 PaaS 资源（如缓存，数据库，消息）和 IaaS 资源（计算、存储、网络，以及容器）；而且为了提升业务的交付速度，软件架构会越来越重度依赖 PaaS 和 IaaS 服务。

2. Infrastructure as Code，就是用代码化的方式，定义及管理 PaaS 资源和 IaaS 资源，进而提升 reusability, consistency, transparency。

# Infrastructure as Code 的优势

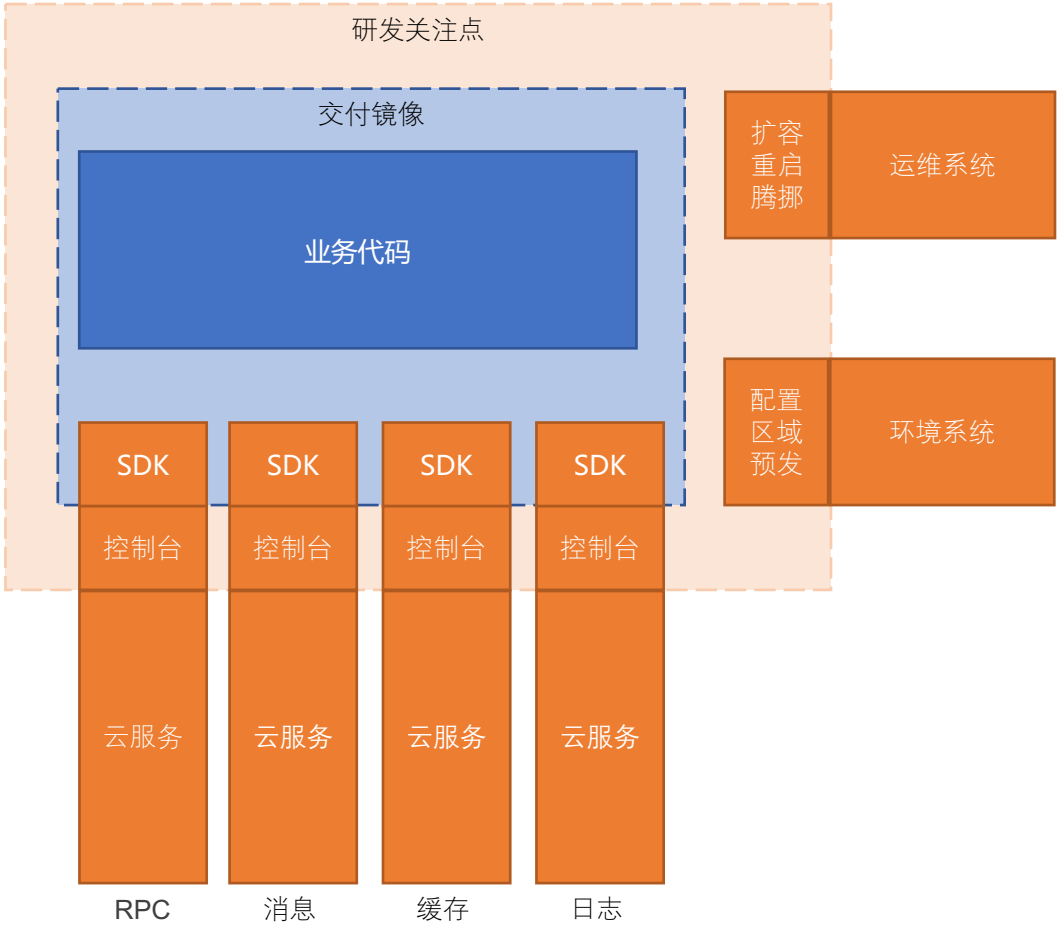
1. 使用 PaaS/IaaS 实现更快的业务交付（通常使用云服务）。
2. 更安全地变更基础设施，降低风险。
3. 安全管控、合规管控更方便。
4. 使用各类基础设施有一致的体验（云上，云下，甚至不同的云）。

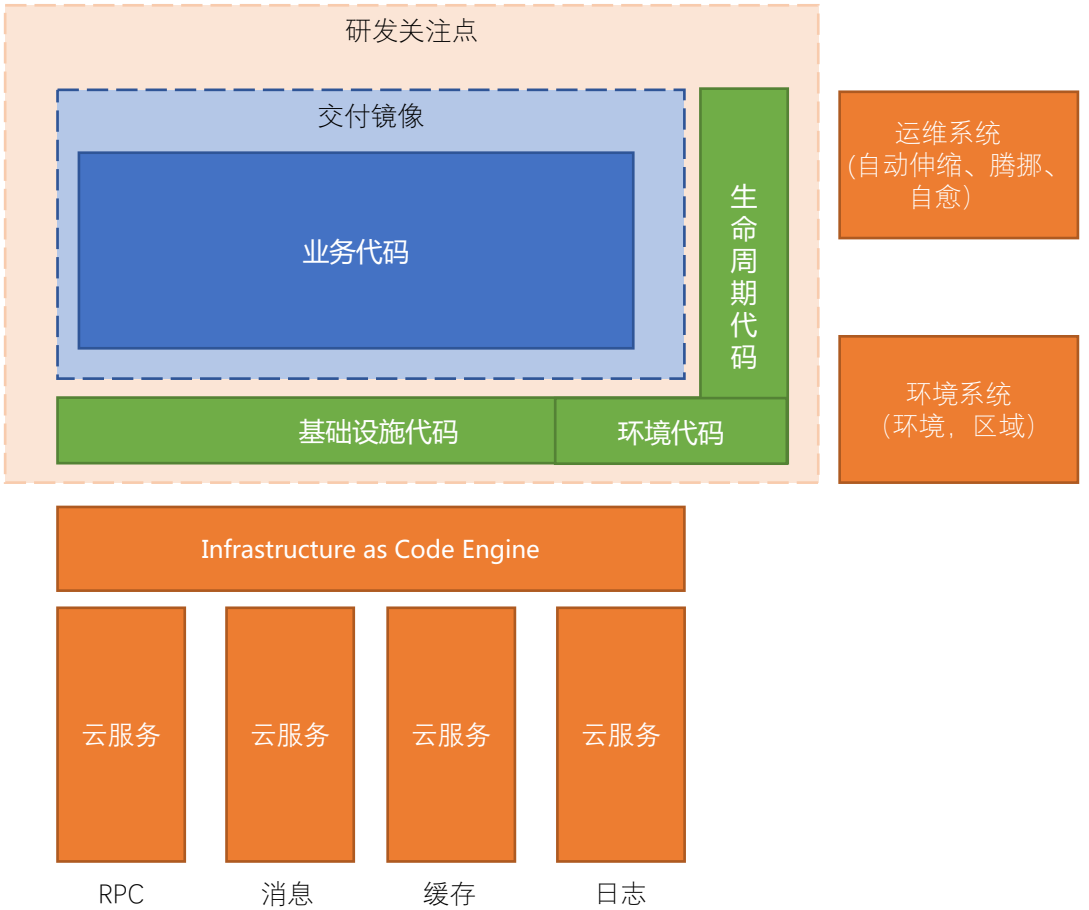
# 阿里巴巴的一些现实情况

1. 已经实现了 IaaS 全面上云，使用阿里云提供的容器资源。
2. 云原生化在进行中，逐步用云上的中间件服务替换集团内的中间件服务。
3. 存在大量，风格各异，体验不一的资源管控控制台。
4. 有比较严格的安全生产要求，以及相应的严格的变更管控流程。
5. 各个 BU 都存在需求各异的架构治理诉求。


我们认为 Infrastructure as Code 是实现云原生的关键技术

# Infrastructure as Code 带来的应用架构变化



# 简单的 Use Case

- 以下的 use case 全部使用 cuelang 描述，关于 cuelang 可以参考 https://cuelang.org/

## Use Case: 服务模版

```
#Service: serverless.#Service & {
    mainContainer:   container.#Main
    sidecar?:        *_|_ | _|_
    resource:        *res.#Medium | res.#Medium | res.#Large | res.#LargeX
    lifeCycle:       lifecycle.#Data
    terminationGracePeriodSeconds: *60 | int & >0
    timezone:        "America/Los_Angeles" | "Asia/Shanghai"

    staticConfig?:   [...staticconfig.#Schema]
    logCollecting?:  logging.#Schema
}
```

# Use Case: 容器规格

```
#Schema: {
    cpu:    *1 | int & >=1 & <=512                                                  // cpu
    memory: *"2048Mi" | =~"^([1-9][0-9]{0,63})(E|P|T|G|M|K|Ei|Pi|Ti|Gi|Mi|Ki)$" // memory
}


#Tiny: #Schema & {
    cpu:    1
    memory: "1Gi"
}

#Small: #Schema & {
    cpu:    1
    memory: "2Gi"
}

#Medium: #Schema & {
    cpu:    2
    memory: "4Gi"
}

#Large: #Schema & {
    cpu:    4
    memory: "8Gi"
}

#LargeX: #Schema & {
    cpu:    8
    memory: "16Gi"
}
```

# Use Case: 发布策略

```
#Schema: #Canary

#Canary: {
    type: "Canary"

    betaReplicas?: int & >0
    stepWeight:    int & >0
    pausePolicy:   *"firstPause" | "manual" | "firstPause" | "auto"
    interval:      *"60s" | time.Duration
    service?: {
        name:       =~"^[a-zA-Z0-9./\\\\?-]{1,255}$"
        port:       *80 | int & >0 & <65535
        targetPort: int & >0 & <65535
    }
}


#Normal: releasestrategy.#Schema & {
    stepWeight:  50
    pausePolicy: "firstPause"
    interval:    "120s"
}


#Core: releasestrategy.#Schema & {
    stepWeight:  16
    pausePolicy: "firstPause"
    interval:    "180s"
}
```

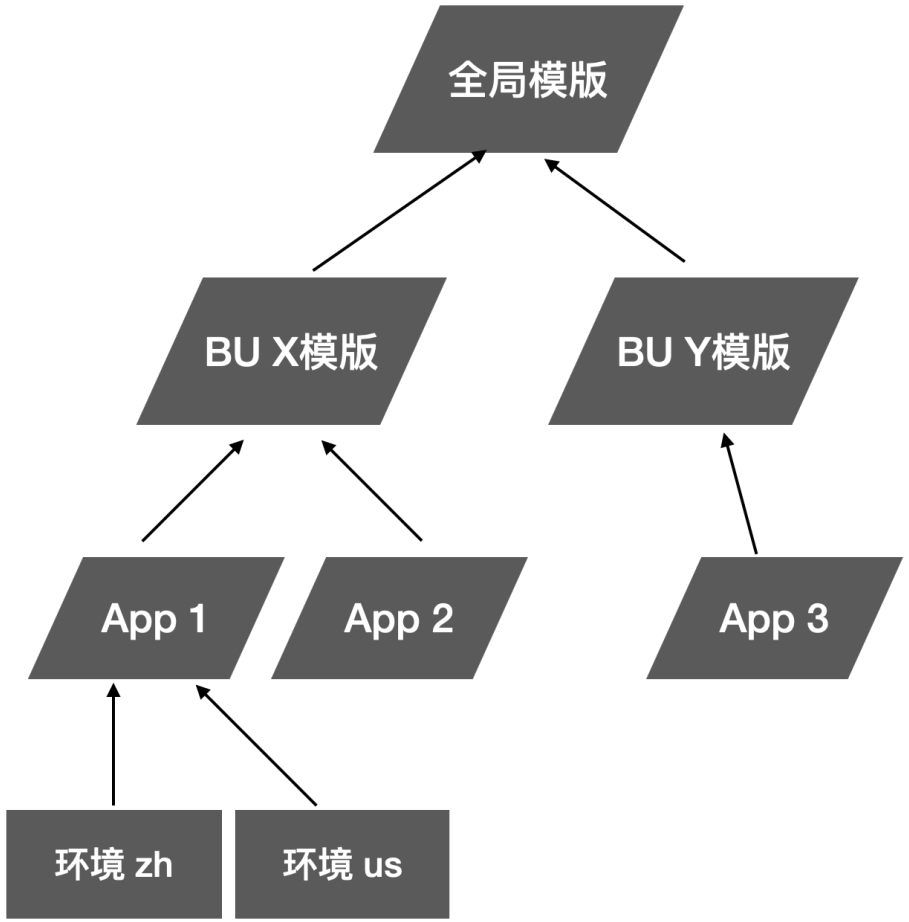
# Use Case: 日志采集

```
#ProductionStdout: {
    slsRegionID:   "*"
    slsProject:    "*"
    slsLogstore:   "*"
}


#ProductionLogFile: {
    slsRegionID:   "*"
    slsProject:    "*"
    slsLogstore:   "*"
}

#LogCollecting: #Schema & {
    stdout: #ProductionStdout
    files: [
        #OnlineLogFile & {
            dir:         "~/logs/boot"
            filePattern: "*.log"
        },
        #OnlineLogFile & {
            dir:         "~/logs/_sls_"
            filePattern: "*.log"
        },
        #OnlineLogFile & {
            dir:         "~/logs/elk"
            filePattern: "*.log"
        }
    ]
}

```

# Use Case: 探针

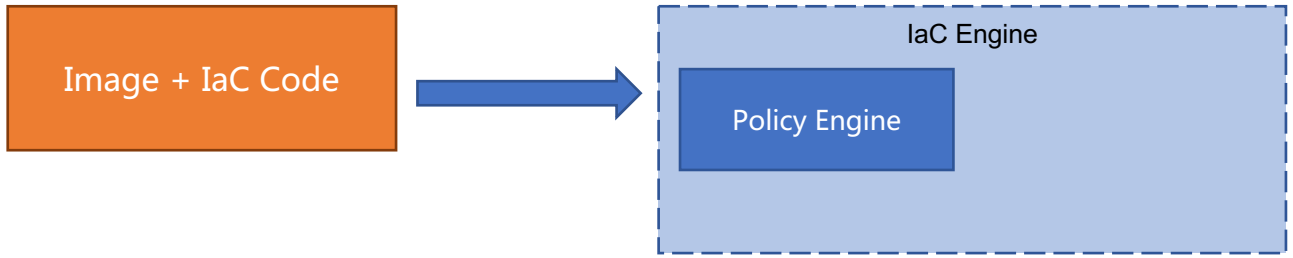
```
#Schema: {
    #handler

    initialDelaySeconds: int32
    periodSeconds:       int32 & >0
    timeoutSeconds:      int32 & >0
    successThreshold:    int32 & >0
    failureThreshold:    int32
}

#handler: #exec | #http | #tcp

#exec: {
    type: "exec"
    exec: [...string]
}


#LivenessProbe: probe.#Schema & {
    exec: ["~/liveness.sh"]

    initialDelaySeconds: 1800
    periodSeconds:       5
    timeoutSeconds:      1
    successThreshold:    1
    failureThreshold:    3
}

```

# IaC code 的复用




1. 多层级的继承，方便复用。
2. 语言层面的扩展限制，保证子层级无法破坏父层级限制，确保安全性。
3. 计划加入模块版本和管控能力，进一步提供架构治理能力。

# Policy as Code (基于 Open Policy Engine)




控制重启过多容器的风险:

```
check_reboot_counts_pod[msg] {
    input.resourceType = "pod"
    rate := input.objCounts / input["trait-manual-scaler"].properties.replicaCount
    rate > 0.5
    msg := sprintf("Pod重启比例不能超过 %v", [rate])
}
```

控制分批观察时间不够的风险:

```
check_batch[msg] {
    waitMinutes := 30
    input.currentBatch > 1
    input.passedMinutes > 0
    input.passedMinutes < waitMinutes
    msg := sprintf("第1批后要求每批间隔%v分钟, 目前剩余: %v 分钟",
[waitMinutes, waitMinutes - input.passedMinutes])
}
```

控制不小心缩容至零的风险:

```
check_replica_zero[msg] {
    input["trait-manual-scaler"].properties.replicaCount == 0
    msg := "不允许缩零"
}
```

# Infrastructure as Code 面向 SRE 的一些场景

1. 架构分析：全局的分析应用使用的软件版本，服务版本，具体的用量等。
2. 批量操作：使用代码批量重启/清理日志/扩缩容。
3. 架构升级：全局升级软件版本，服务版本等，以降低服务成本，降低安全风险。
4. 管控：更丰富的 Policy。
5. 建站：自动化新建站点。

reusability, consistency, transparency

# Infrastructure as Code 下一步

1. 工具生态的建设，如 IDE Plugin，命令行工具
2. 和阿里云服务的深度集成，丰富的 live diff 能力等
3. 围绕各类场景积累丰富的 module
4. IaC Engine 的开源?

# THANKS

juven.xuxb@alibaba-inc.com