名词：

服务发现：

用来确保服务的位置无关性，通过服务名来查询获得服务的实际地址。

名字解析：

用来确保服务器位置无关性，通过机器名查询获得机器的实际IP地址。

场景一：

特点：

应用少，流量轻，数台机器，DevOps分离，手动基础设施管理，手动应用程序部署，监控非必要，面向机器，基于IP的配置，服务发现非必要，名字解析非必要。

流程：

DevOps分离。

方法：

各个环境的基础设施（网路，服务器，IP地址，操作系统，依赖包）事先由Ops人员搭建，完成后将各个环境最终的机器配置信息告知Dev。机器配置信息包括机器名，用途，规格（OS，CPU，内存，硬盘），IP地址，端口，预装软件信息等。

Dev开发完一个迭代后，将软件打包后交由Ops人员部署到QA环境，一旦QA环境测试通过再由Ops人员将软件包部署到PROD环境。Dev要根据Ops人员提供的各个环境的机器配置信息生成应用在各个环境的具体配置文件，并随软件包一起提交给Ops人员。

工具：

手动部署

场景二：

特点：

应用少，流量轻，数台机器，DevOps合作，手动基础设施管理，自动应用程序部署，监控非必要，面向机器，基于IP的配置，服务发现非必要，名字解析非必要。

流程：

DevOps合作。

方法：

各个环境的基础设施（网路，服务器，IP地址，操作系统）事先由Ops人员搭建，完成后将各个环境最终的机器配置信息告知Dev。机器配置信息包括机器名，用途，规格（OS，CPU，内存，硬盘），IP地址，端口，预装软件信息等。

Dev提交一个Commit到Git Repo后触发CI/CD执行Pipeline的第一个Stage Build，Build Stage从GitRepo下载代码（包含部署脚本）并编译，测试，打包，上传。

打包完成后自动触发CI/CD执行Pipeline的第二个Stage QA，QA Stage调用部署程序（Ansible）根据部署脚本（Playbook）将软件包部署到QA环境（Inventory），部署通常包括下载软件包，根据Inventory生成配置文件，部署程序，启动服务等。

一旦QA环境测试通过，需手动触发CI/CD执行Pipeline的下一个Stage Prod，Prod Stage调用部署程序（Ansible）根据部署脚本（Playbook）将软件包部署到Prod环境（Inventory）。

工具：CI/CD + Ansible

CI/CD- GoCD Pipeline：

Build
QA
Prod

部署脚本- Ansible：

```
invetories
    qa
        [web]
        ......
        [db]
        ......
```

```
        prod
            [web]
            ......
            [db]
            ......
    playbooks
        web.yml
        db.yml
    templates
        Conf.j2
```

场景三：

特点：

应用少，流量轻，数台机器，DevOps合作，自动基础设施管理，自动应用程序部署，监控非必要，面向机器，基于IP的配置，服务发现非必要，名字解析非必要。

流程：

DevOps合作。

方法：

各个环境的基础设施（网络，服务器，IP地址，操作系统）由基础设施管理程序（Terraform）根据各个环境的基础设施配置自动配置，完成后自动生成各个环境最终的机器配置信息（Invetory）。

Dev提交一个Commit到Git Repo后触发CI/CD执行Pipeline的第一个Stage Build，Build Stage从GitRepo下载代码（包含部署脚本）并编译，测试，打包，上传。

打包完成后自动触发CI/CD执行Pipeline的第二个Stage QA，QA Stage调用基础设施管理程序（Terraform）根据基础设施配置自动完成QA环境基础设施配置并生成QA环境机器配置信息（Inventory）。再由部署程序（Ansible）根据部署脚本（Playbook）将软件包部署到QA环境（Inventory），部署通常包括下载软件包，根据Inventory生成配置文件，部署程序，启动服务等。

一旦QA环境测试通过，需手动触发CI/CD执行Pipeline的下一个Stage Prod，Prod Stage调用基础设施管理程序（Terraform）根据基础设施配置自动完成Prod环境基础设施配置并生成Prod环境机器配置信息（Inventory）。再由部署程序（Ansible）根据部署脚本（Playbook）将软件包部署到Prod环境（Inventory）。

工具：CI/CD + Terraform + Ansible

CI/CD- GoCD Pipeline：

Build

QA

Prod

基础设施配置- Terraform：

input：

qa：

*.tf

*.tf_state

prod：

*.tf

*.tf_state

output: inventories

qa

prod

部署脚本- Ansible：

invetories

qa - created from terraform

prod - created from terraform

playbooks

web.yml

db.yml

templates

Conf.j2

场景四：

特点：

应用多，流量巨，万台机器，DevOps合作，自动基础设施管理，自动应用程序管理，监控，面向容器，服务发现，名字解析，配置管理。

流程：

DevOps合作。

方法：

各个环境的基础设施（网络，服务器，IP地址，操作系统）由基础设施管理系统根据需求信息（由高层提供）自动配置。各个服务的用途也由基础设施管理系统根据需求信息（由高层提供）自动决策。因基础设施并非事先设定，故应用的配置不能依赖服务和机器的物理地址，需采用服务发现配合名字解析来实现系统的集成和配置。

工具：

自动基础设施管理系统（面向机器，伸缩，部署，监控）
自动应用程序管理系统（面向容器，伸缩，部署，监控）
服务发现
名字解析
配置管理