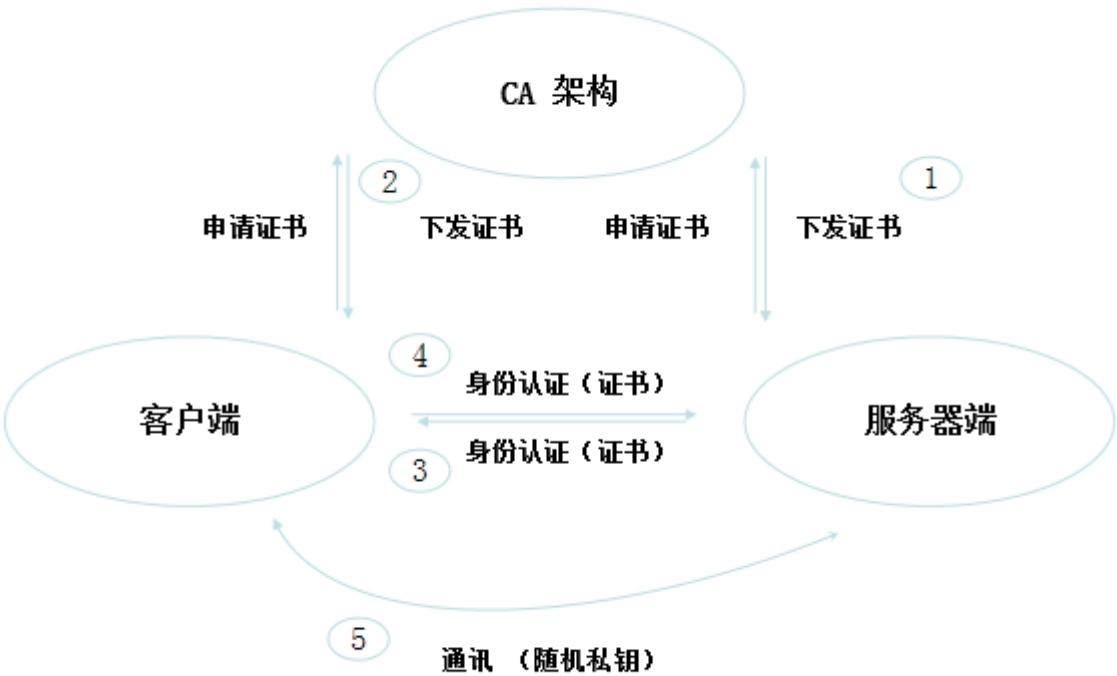
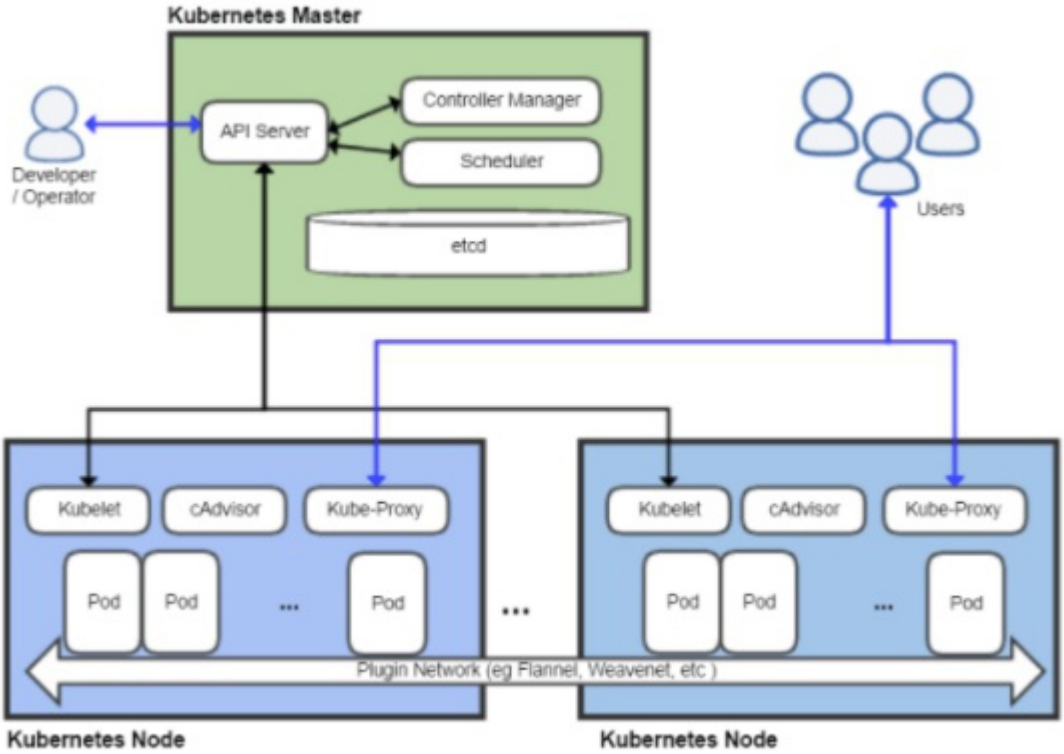
Authentication

- HTTP Token 认证：通过一个 Token 来识别合法用户
  - HTTP Token 的认证是用一个很长的特殊编码方式的并且难以被模仿的字符串 - Token 来表达客户的一种方式。Token 是一个很长的很复杂的字符串，每一个 Token 对应一个用户名存储在 API Server 能访问的文件中。当客户端发起 API 调用请求时，需要在 HTTP Header 里放入 Token
- HTTP Base 认证：通过 用户名+密码 的方式认证
  - 用户名+：+密码 用 BASE64 算法进行编码后的字符串放在 HTTP Request 中的 Heather Authorization 域里发送给服务端，服务端收到后进行编码，获取用户名及密码
- 最严格的 HTTPS 证书认证：基于 CA 根证书签名的客户端身份认证方式

Ⅰ、HTTPS 证书认证：



Ⅱ、需要认证的节点



两种类型

- **Kubenetes 组件对 API Server 的访问：kubectl、Controller Manager、Scheduler、kubelet、kube-proxy**
- **Kubernetes 管理的 Pod 对容器的访问：Pod（dashborad 也是以 Pod 形式运行）**

安全性说明

- **Controller Manager、Scheduler 与 API Server 在同一台机器，所以直接使用 API Server 的非安全端口访问，** `--insecure-bind-address=127.0.0.1`
- **kubectl、kubelet、kube-proxy 访问 API Server 就都需要证书进行 HTTPS 双向认证**

证书颁发

- **手动签发：通过 k8s 集群的跟 ca 进行签发 HTTPS 证书**
- **自动签发：kubelet 首次访问 API Server 时，使用 token 做认证，通过后，Controller Manager 会为 kubelet 生成一个证书，以后的访问都是用证书做认证了**

Ⅲ、kubeconfig

kubeconfig 文件包含集群参数（CA证书、API Server地址），客户端参数（上面生成的证书和私钥），集群 context 信息（集群名称、用户名）。Kubenetes 组件通过启动时指定不同的 kubeconfig 文件可以切换到不同的集群

Ⅳ、ServiceAccount

Pod中的容器访问API Server。因为Pod的创建、销毁是动态的，所以要为它手动生成证书就不可行了。Kubenetes使用了Service Account解决Pod 访问API Server的认证问题

Ⅴ、Secret 与 SA 的关系

Kubernetes 设计了一种资源对象叫做 Secret，分为两类，一种是用于 ServiceAccount 的 service-account-token， 另一种是用于保存用户自定义保密信息的 Opaque。ServiceAccount 中用到包含三个部分：Token、ca.crt、namespace

- **token是使用 API Server 私钥签名的 JWT。用于访问API Server时，Server端认证**
- **ca.crt，根证书。用于Client端验证API Server发送的证书**
- **namespace, 标识这个service-account-token的作用域名空间**

```
kubectl get secret --all-namespaces
kubectl describe secret default-token-5gm9r --namespace=kube-system
```

默认情况下，每个 namespace 都会有一个 ServiceAccount，如果 Pod 在创建时没有指定 ServiceAccount，就会使用 Pod 所属的 namespace 的 ServiceAccount

总结