学习内容：
Go 程序设计语言的第十章，云原生公开课第八、九、十章
收获：

- 空导入可以只加载包里的初始化代码
- 可以给导入的包取别名来避免冲突
- 环境变量 GATH 来保存工作空间的地址
- go install 与 build 命令相比，能够保存中间代码，让下次编译加速
- GOOS 与 GOARCH 环境变量用来表示编译的目标环境
- 可以通过 go 文件的文件名后半段来告诉编译器自己的目标环境
- 可以通过将 package 封装在 internal 文件夹里让只有几个 package 可以看见
- 使用 go list 配合…来查询包
- Pod 配置管理:
    1. 可变配置就用 ConfigMap
    2. 敏感信息是用 Secret
    3. 身份认证是用 ServiceAccount 这几个独立的资源来实现的
    4. 资源配置是用 Resources
    5. 安全管控是用 SecurityContext
    6. 前置校验是用 InitContainers 这几个在 spec 里面加的字段，来实现的这些配置管理
- Pod Volumes 的常见类型：
    1. 本地存储，常用的有 emptydir/hostpath；
    2. 网络存储：网络存储当前的实现方式有两种，一种是 in-tree，它的实现的代码是放在 K8s 代码仓库中的，随着 k8s 对存储类型支持的增多，这种方式会给 k8s 本身的维护和发展带来很大的负担；而第二种实现方式是 out-of-tree，它的实现其实是给 K8s 本身解耦的，通过抽象接口将不同存储的 driver 实现从 k8s 代码仓库中剥离，因此 out-of-tree 是后面社区主推的一种实现网络存储插件的方式；
    3. Projected Volumes：它其实是将一些配置信息，如 secret/configmap 用卷的形式挂载在容器中，让容器中的程序可以通过 POSIX 接口来访问配置数据；
    4. PV 和 PVC
- PVC 像是接口，PV 像实现
- PV 的生产方式：静态 Provisioning、动态 Dynamic Provisioning
- PV->StorageClass->PV;
  VolumeSnapshot->VolumeSnapshotClass->VolumeSnapshotContent
- Local PV 通过 nodeAffinity 来指定 PV 的拓扑域
- Dynamic Provisioning PV 通过指定 StorageClass 的 allowedTopologies 来指定创建的 PV 的拓扑域