EC2:

**Amazon Machine Image(AMI):**

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0b9064170e32bde34 (64-bit x86) / ami-026141f3d5c6d2d0c (64-bit Arm)

**Instance Type:**

t2.micro

**Storage:**

8GB

**Security Group:**

HTTP Port 80 Source: Anywhere

Custom TCP Port 8080 Source: Anywhere

SSH Port 22 Source: Anywhere

HTTPS Port 443 Source: Anywhere

**Launch and create a key pair**

**Key Operation:**

1. ls | grep keyname.pem

2. sudo chmod 0600 keyname.pem

3. ssh-add keyname.pem

4. ssh ubuntu@<YOUR\_IP\_ADDRESS> (IPV4 address of the EC2 Instance, for example, 18.222.25.78)

3. ssh -i keyname.pem [ubuntu@server](mailto:ubuntu@server) (server is the IPV4 DNS of the EC2 Instance, for example: ec2-18-222-25-78.us-east-2.compute.amazonaws.com (our first instance))

4. sudo apt update

5. sudo apt install python3 python3-pip tmux htop

6. mkdir Apolloapp

7. cd Apolloapp

Some Bugs:

**OSError: [Errno 98] Address already in use**

ps -fA | grep python

kill kidnum

7. pip freeze > requirements.txt (local terminal)

Vue(MVVM):

node -v

npm install -g @vue/cli

vue create projectname

Flask(MVC):

多考虑: extension

jinja2 模板引擎

Werkzeug WSGI 工具集

Itsdangerous 基于Django的签名模块

python3.6 -m venv env

source env/bin/activate (env)

pip install Flask==1.1.2 Flask-Cors==3.0.10

python prpjectname.py (运行)

GIL锁 python有全局锁，性能低, 锁到1/cpu核心数, 所以开cpu核心数个进程

app.run(debug=True, port=8000, host=’0.0.0.0’)

插件，扩展库：

1. 帮助开发者，快速实现某种功能

2. 使用流程

下载，安装

初始化配置

flask-script:

可以帮助开发者快速实现命令行参数接收

pip install flask-script

manager = Manager(app=app)

启动：

manager.run()

初始化:

使用app构建manager对象

使用manager 启动程序

使用:

shell

进入flask的环境的shell

runserver

启动服务器

可以指定参数

-d调试模式 -r 是否重新加载文件 -h 指定主机 -p 指定端口 -threaded 是否使用多线程

python projectname.py runserver -d -r -h 0.0.0.0 -p 9000

参数:

路径参数

位置参数

关键字参数 <int: name>提取

请求参数

get参数在路径中?之后

post参数在请求体

flask中参数

都是关键字参数

默认标识是尖括号<name>

name需要和对应的视图函数的参数名字保持一致

参数允许有默认值

如果有默认值，那么在路由中，不传输参数也是ok的

如果没有默认值，参数在路由中必须传递

参数语法:

converter:var

string: 斜线是参数分隔符

path： 将斜线认为是字符

返回字符串

返回response: reponse = make\_response()

jsonify 将数据格式化为json格式，同时设置返回类型为application/json

json.dumps将数据格式化为json格式，没有设置返回的数据类型，默认类型是text/html

pip install flask-blueprint

会话技术：

想跨越请求传递数据

web开发中使用的短连接

cookie

客户端会话技术

数据都是存储在浏览器中

支持过期

不能跨域

frame 标签

可以直接加载整个网站

不能跨浏览器

cookie 是通过response来进行操作

flask中的cookie可以直接支持中文

flask对cookie中的内容做了编码

session

服务端会话技术

对数据进行数据安全操作

默认在内存中

不容易管理

容易丢失

不能多台电脑写作

flask-session 默认有效期31天

flask-session 插件(放数据库里)：

pip install flask-session

token

直接把消息闪到页面上flash()

get\_flashed\_messages()

editconfig (格式)

AWS

RESTful API接口

SDK开发工具包

Command Line Interface 命令行工具

Management Console 管理控制台

IAM 安全

elastic beanstalk 运行和管理web应用程序

**怎么处理静态内容？**

S3服务 （存储静态资源）

S3(Simple Storage Service) 99.999999999% 可靠性

最多支持5T大小的单个对象 存储无限量

存储桶

上传静态内容，获取链接， 加入到html等（要不是private, 设置权限）

**我的服务器哪里来？**

EC2(Elastic Cloud Compute)

容量可以根据需求变化

一些种类： 常规通用 计算优化 内存优化 存储优化

**创建EC2**

实例

系统映像

实例类型

配置实例的信息（网络）

用户数据（启动就提供外部服务）（公有ip）

添加存储

添加标签

安全相关（现有）

停止 =》 调节大小

**多个服务器并行工作？**

负载均衡（Elastic Load Balancing） ELB,

1. 做流量分发

2. 后端实例的健康检查

3. 提供现在的参数指标供后续的使用

**创建负载均衡器**

按流程

**我有多少客户？能否弹性的获得资源？**

AutoScaling 自动增加EC2的资源 自动的收缩

CloudWatch 监控服务 （ELB=>CloudWatch(对收集到的数据做警报，根据自己设定的阈值)=>AutoScaling）

弹性伸缩服务: ELB CloudWatch AutoScaling (大规模高并发)

EC2 Container Service （类似于docker）

同时运行web, 计算

EBS (Elastic Block Store)

数据与计算实例相分离 I/O密集型

CloudFront 降低延迟？

RDS relational database service

VPC virtual private cloud

Elastic Beanstalk 简单快速web,可以基于代码或者容器，网页三层

CodeDeploy => ec2

docker => 虚拟的集群=> ECS(Elastic Container Service)