1. 客户业务场景描述

金蝶苍穹平台包含金蝶的aPaaS平台 + 通用管理平台GPaaS平台。其上不仅支撑金蝶自行开发的面向大、中、小微的水平ERP SaaS服务；同时还面向众多金蝶系子公司和第三方开发团队提供PaaS支撑能力，向行业纵深方向成长垂直SaaS服务。因此，培养开发者生态，提升开发者效率，是金蝶战略目标之一。

而当前，金蝶的开发者生态为满足日常的开发和测试需要，通常需要在本地自行搭建一套完整的paas平台和开发环境，搭建周期长达数周，资源浪费效率低下。为了解决此问题，苍穹启动云上多租户开发/测试环境部署。

POC目标：

1. 为金蝶的SaaS ISV提供云端共享的开发环境。ISV只需登录开发者平台，即可在云端可获得IDE开发及金蝶苍穹平台测试环境；
2. 开发者平台环境应支持多租户模式，计费独立且运营费用较低
3. 尽量使用托管服务，减少运维。
4. 客户对成本非常敏感。金蝶目前已有800ISV，按照每个ISV平均10个开发者计算，将有8000用户。

2. POC架构设计 & 涉及的Service

POC分3阶段进行：

- IDE环境选择及功能测试
- 苍穹平台对EKS的适配测试
- 多租户模式下的计费方案测试

2.1 IDE环境选择

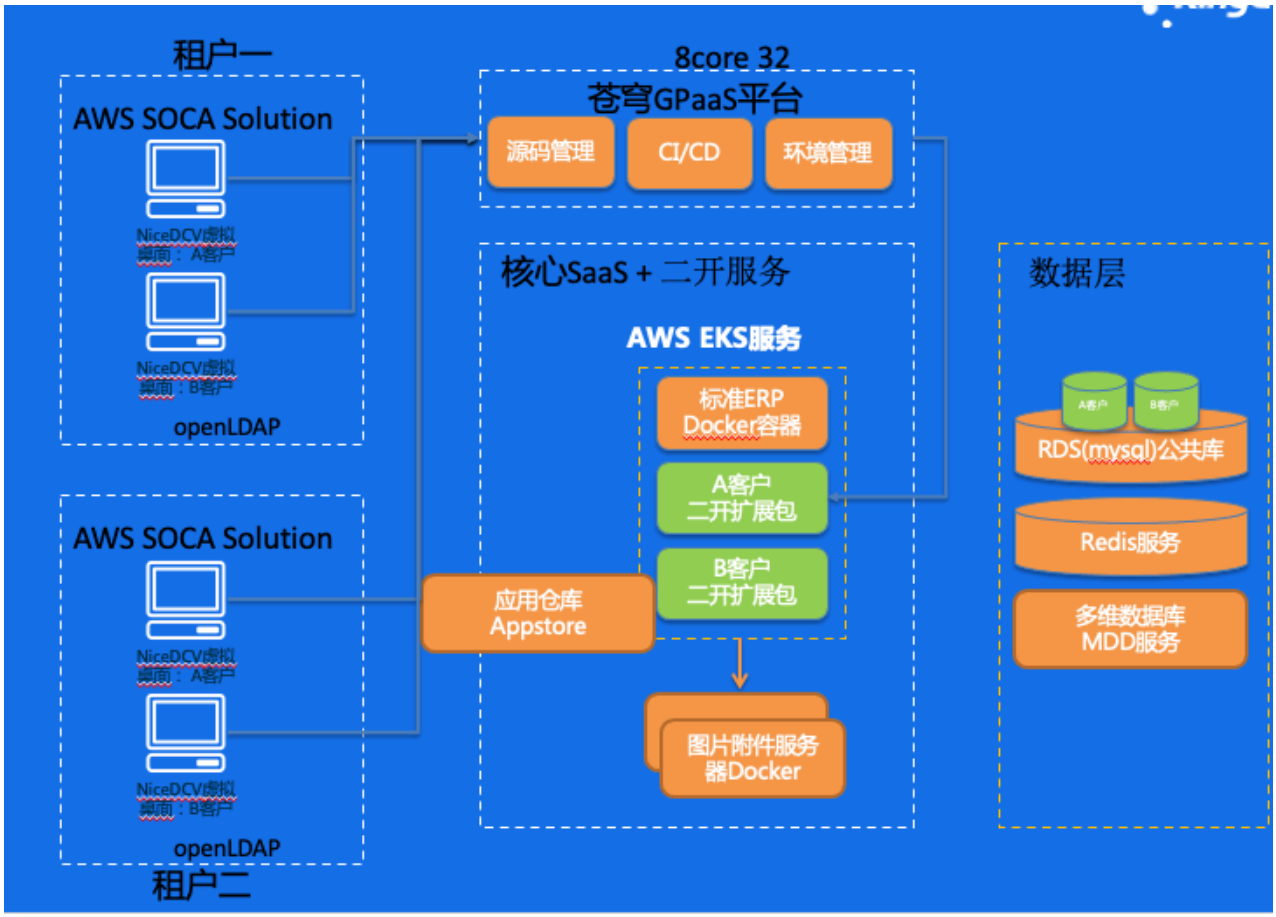
先后测试了3个IDE环境：

1. Cloud9：WebIDE环境，支持Java开发。中国区未上线，使用marketplace的cloud9
2. Soca：sacle-out compute on AWS solution. 提供NiceDCV虚拟桌面，构建支持eclipse的预装开发环境
3. Theia：开源的webIDE环境，支持java开发。

测试项	cloud9	soca	Theia
工具特点	WebIDE环境	提供NiceDCV虚拟桌面	WebIDE环境
多租户环境支持	难以脱离aws console提供管理	每ISV对应一个单独的Soca环境。易于集成	提供容器化部署，易与苍穹基于K8S的多租户部署模式集成
开发环境支持	不支持：代码自动补足：java反编译及断点功能	安装完整的Eclipse软件，符合开发者使用体验	不支持：java反编译及断点功能
开发体验	相对简单，但java反编译及断点功能缺失。	功能完全满足，便于客式化AMI镜像	类Vscode开发界面体验，具有一定的学习成本
与GPaaS环境集成	由于功能缺失，不再验证	天然集成LDAP用户管理，开源基于Python的Django开发框架，便于集成	可通过nginx和外部身份验证体系实现集成
成本评估	由于功能缺失，不再验证	基于vm方式，价格较container高出20%左右	基于容器方式，价格占优

3. 难点解决

3.1 多租户模式及流程设计



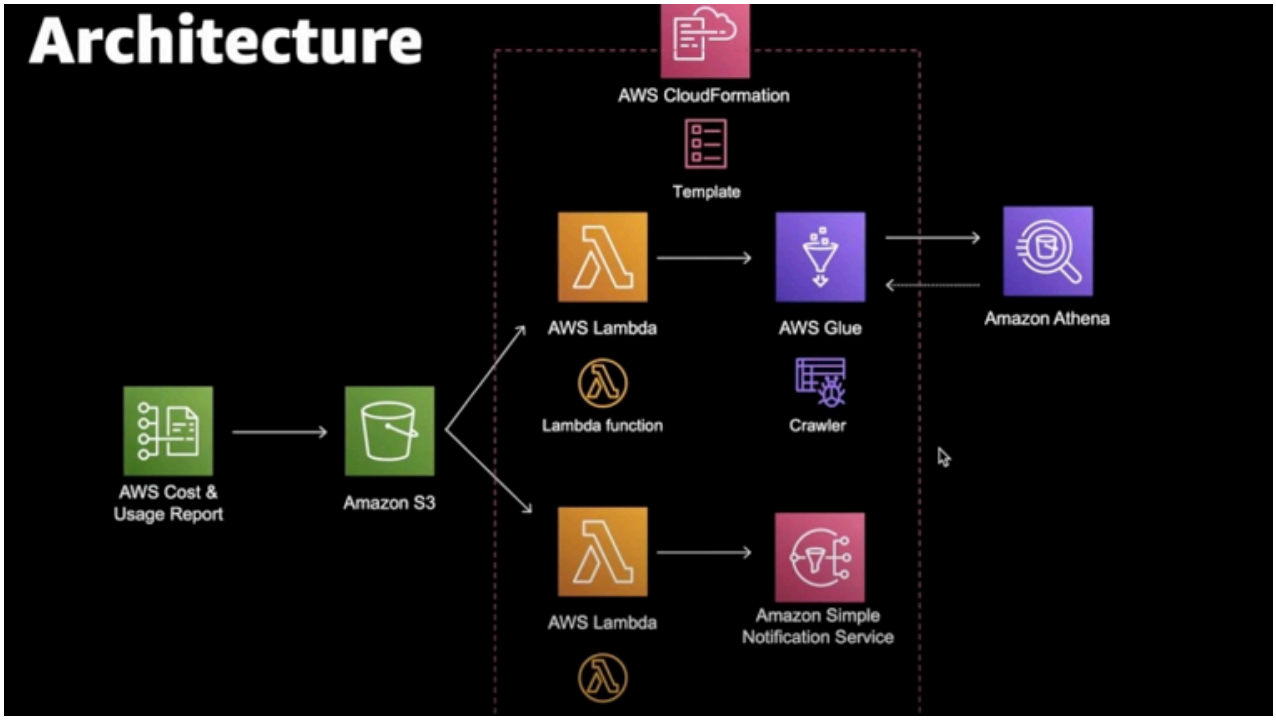
POC中使用soca solution为每个ISV开发商提供基于niceDCV的IDE开发环境，提供ISV的自主的用户管理和费用查看UI。每个ISV支持多开发人员使用各自niceDCV桌面，启动预先构建的开发环境镜像。

所有ISV共享苍穹GPaaS平台和SaaS应用测试环境。借助苍穹已有的CI/CD管道和多租户架构，实现开发包的推送，测试环境的独立部署和测试

3.2 多租户成本考虑及成本监控

Soca solution自身包含opensearch+Kibana用于EDA job的指标与费用监控。项目中对这一部分没有涉及。从成本考虑，删除了cloudformation中的这部分内容。成本监控使用CUR + S3 + Athena方案。

Soca中对每个资源都赋予集群前缀为tag，便于费用的收集与展示。



成本优化：

1. 测试了每个nice DCV的基础配置。从t3large将为t3medium，配置可以满足绝大部分开发需求，成本每用户成本可降低50%。建议提供两种资源模式供用户选择。
2. Soca UI server缺省配置为m5xlarge。对客户当前情况，该server仅需要支持登录、身份验证和虚拟桌面启动，启动后将直接与nice DCV通讯，对soca UI server压力很小。测试后建议客户降配，选择t3large。

3.4 GPaaS平台支持EKS验证

原GPaaS构建在K8S之上，迁移到AWS平台之后，建议客户使用EKS托管服务便利运维。因此对苍穹平台从k8s向eks迁移的角度进行了适配性验证。EKS认证体系与K8S不同，需要用IAM role获取临时安全凭证，再通过安全凭证调用EKS api server。POC中演示了这一过程如下：

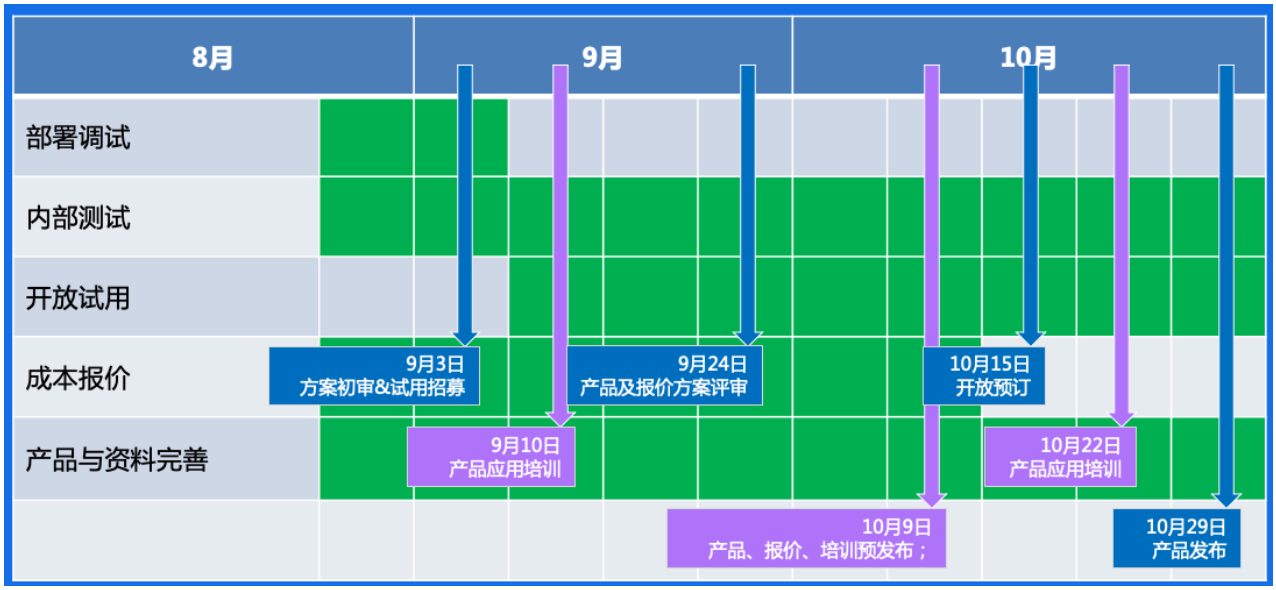
- 根据IAM role通过STS获得具有有效期的临时安全凭证

```python
import json
import base64
import boto3
import re
from botocore.signers import RequestSigner
from kubernetes import client


def get_bearer_token(cluster_id, region):
    STS_TOKEN_EXPIRES_IN = 60
    session = boto3.session.Session(aws_access_key_id="Axxx6PU", aws_secret_access_key="yiipwxxxLG8pMJ")

    client = session.client('sts', region_name=region)
    service_id = client.meta.service_model.service_id

    signer = RequestSigner(
        service_id,
        region,
        'sts',
        'v4',
        session.get_credentials(),
        session.events
    )

    params = {
        'method': 'GET',
        'url': 'https://sts.{}.amazonaws.com.cn/?Action=GetCallerIdentity&Version=2011-06-15'.format(region),
        'body': {},
        'headers': {
            'x-k8s-aws-id': cluster_id
        },
        'context': {}
    }

    signed_url = signer.generate_presigned_url(
        params,
        region_name=region,
        expires_in=STS_TOKEN_EXPIRES_IN,
        operation_name=''
    )

    base64_url = base64.urlsafe_b64encode(signed_url.encode('utf-8')).decode('utf-8')
    return 'k8s-aws-v1.' + re.sub(r'=*', '', base64_url)
```

- 携带临时安全凭证调用EKS API server

```python
def lambda_handler(event, context):
    aToken = get_bearer_token('kingdee', 'cn-northwest-1')
    aConfiguration = client.Configuration()
    aConfiguration.debug = True

    # Specify the endpoint of your Kube cluster
    aConfiguration.host = "https://75058F83314822F453FC876071B20E13.gr7.cn-northwest-1.eks.amazonaws.com.cn:443"


    aConfiguration.verify_ssl = False


    aConfiguration.api_key = {"authorization": "Bearer " + aToken}

    # Create a ApiClient with our config
    aApiClient = client.ApiClient(aConfiguration)

    # Do calls
    v1 = client.CoreV1Api(aApiClient)
    print("Listing pods with their IPs:")
    ret = v1.list_pod_for_all_namespaces(watch=False)
    for i in ret.items:
        print("%s\t%s\t%s" %
              (i.status.pod_ip, i.metadata.namespace, i.metadata.name))

    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

4. POC计划



5. POC takeaways

- 用户的使用体验是决定产品选型的关键
  项目中提供了3中IDE开发环境。但从开发者的使用习惯、环境的响应速度看soca最为适用。同时基于container的web ide --Theia在成本上具有优势。金蝶方案将以两种选择并存的方式提供开发者选择
- CSDC solution在面向客户的方案推荐上具有优势
  较为完整的方案已经集成了客户对界面、niceDCV安装、用户管控等关键需求

  基于cloudformation的快速部署，加速项目的构建和落地。
  同时代码开源便于客户进行客式化，满足个性需求。