



云原生场景下的DevSecOps

郑明 · 2022年12月20日



目录

CONTENTS

1. 云原生面临的新挑战
2. DevSecOps如何应对新挑战
3. 基于免费工具/开源技术打造云原生DevSecOps工具链





1. 云原生面临的新挑战

云原生 (Cloud Native) 是一种构建和运行应用程序的方法，是一套技术体系和方法论。是充分运用云原生能力（自动扩充、无中断部署、自动化管理、弹性等）来进行应用设计、部署和智能化运维的方法。



一、了解云原生

何谓云原生

云原生是面向应用设计的一种**思想理念**，充分发挥云效能的**最佳实践路径**；帮助企业构建**弹性可靠、松耦合、易管理可观测**的应用系统，提升交付效率，降低运维复杂度。代表技术包括容器、微服务、服务网格、不可变基础设施、声明式API及Serverless等。

参考CNCf云原生定义

云原生特点

- 云原生面向应用，而非面向基础设施
- 弹性服务编排（orchestration）
- 开发运营一体化（DevOps）
- 持续集成和持续部署（CI/CD）
- 微服务架构（SOA）
- 无服务模型（Serverless）

微服务的设计和无服务的功能是云原生理念的核心体验！

容器、编排、服务网格均是实现云原生的关键技术！



二、了解云原生技术

不可变基础设施

部署完成后基础设施不会被修改。如需更新，则更改镜像构建新服务以替代旧服务，减少或杜绝可变基础设施中如配置偏移、集群配置一致性的问题。



服务网格

将应用与通用功能解耦，实现应用轻量化，使开发者无需关注微服务相关治理问题，聚焦业务逻辑本身。目前最主要的开源项目Istio.



容器

通过对应用打包和隔离，让应用在不同环境之间轻松迁移，可以实现秒级部署，同时容器应用易于移植，一次构建，随处部署。



云原生 关键技术



声明式API

减少开发人员工作量，让分布式系统之间的交付变得简单。



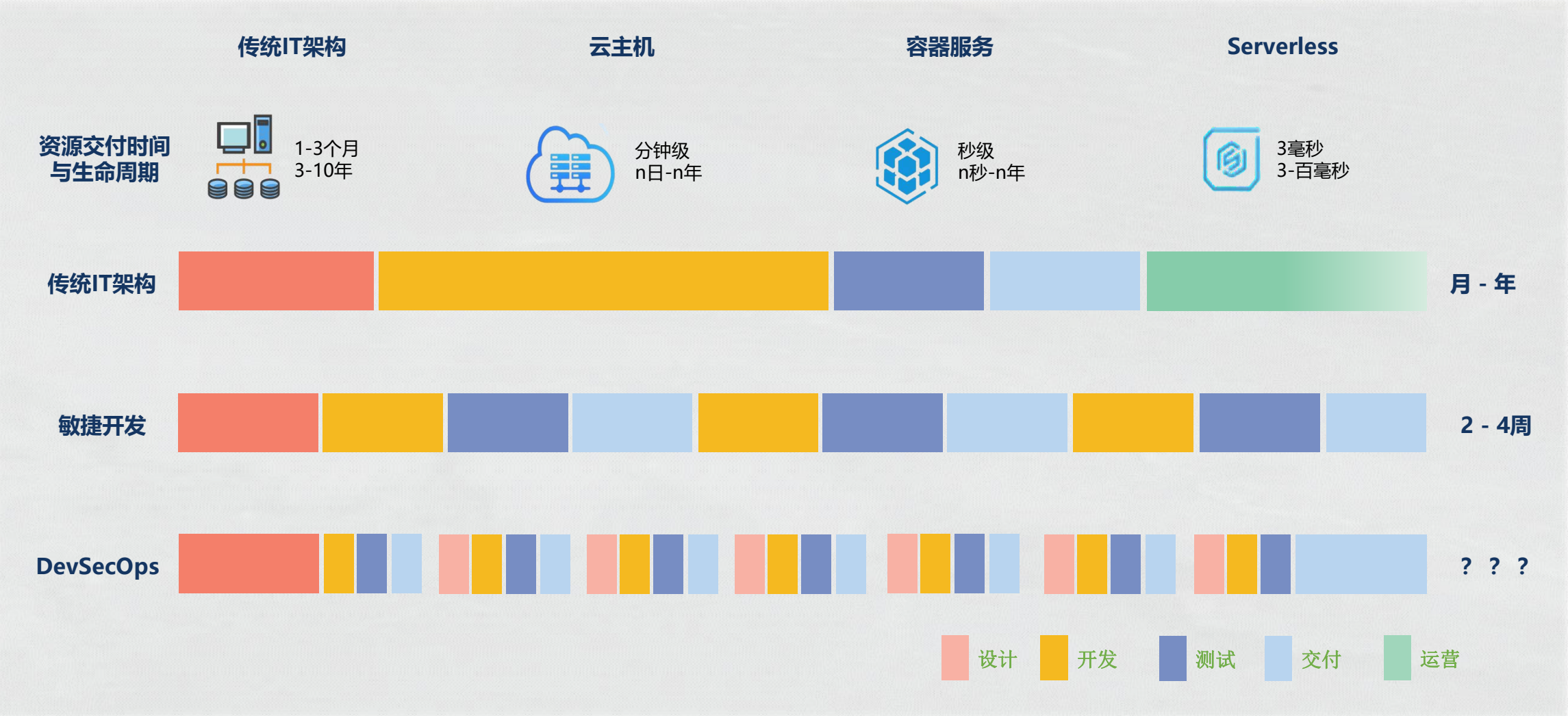
容器

将单体应用拆分多个子应用，每个子应用负责一组子功能，便于简化与加速应用的迭代过程，同时也能提高应用可观测性，可容错性等特性。

参考来源：CNCF，沙利文整理



三、云原生颠覆了IT生命周期和治理模式的变迁



相关统计显示：接近**50%**的容器服务生命周期**小于5分钟**；
将近**70%**生命周期**小于1小时**。



四、云原生场景下云计算服务模式（XaaS）

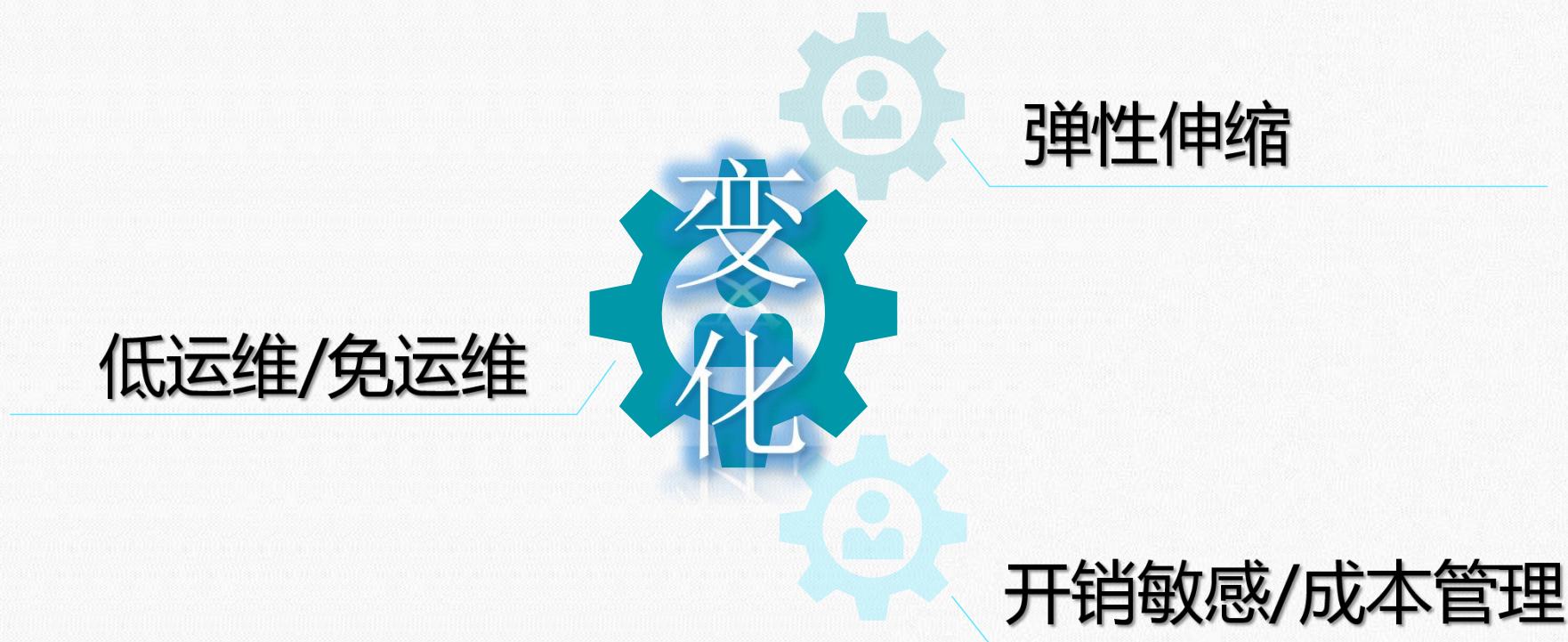


云原生堆栈可以使用不同的部署模型（IaaS、PaaS、CaaS 和 FaaS、SaaS）来部署。其中IaaS、PaaS、SaaS已是众所周知并且使用多年，其中CaaS（容器即服务）和FaaS（功能即服务）为云原生的特有模型。CaaS模型可以让用户通过利用基于容器的虚拟化平台、应用编程接口（API）或Web 门户管理接口来编排和管理容器、应用和集群。FaaS模型可以让企业用户通过执行代码对事件作出响应，而无需关心通常与构建和启动微服务相关的复杂基础架构。

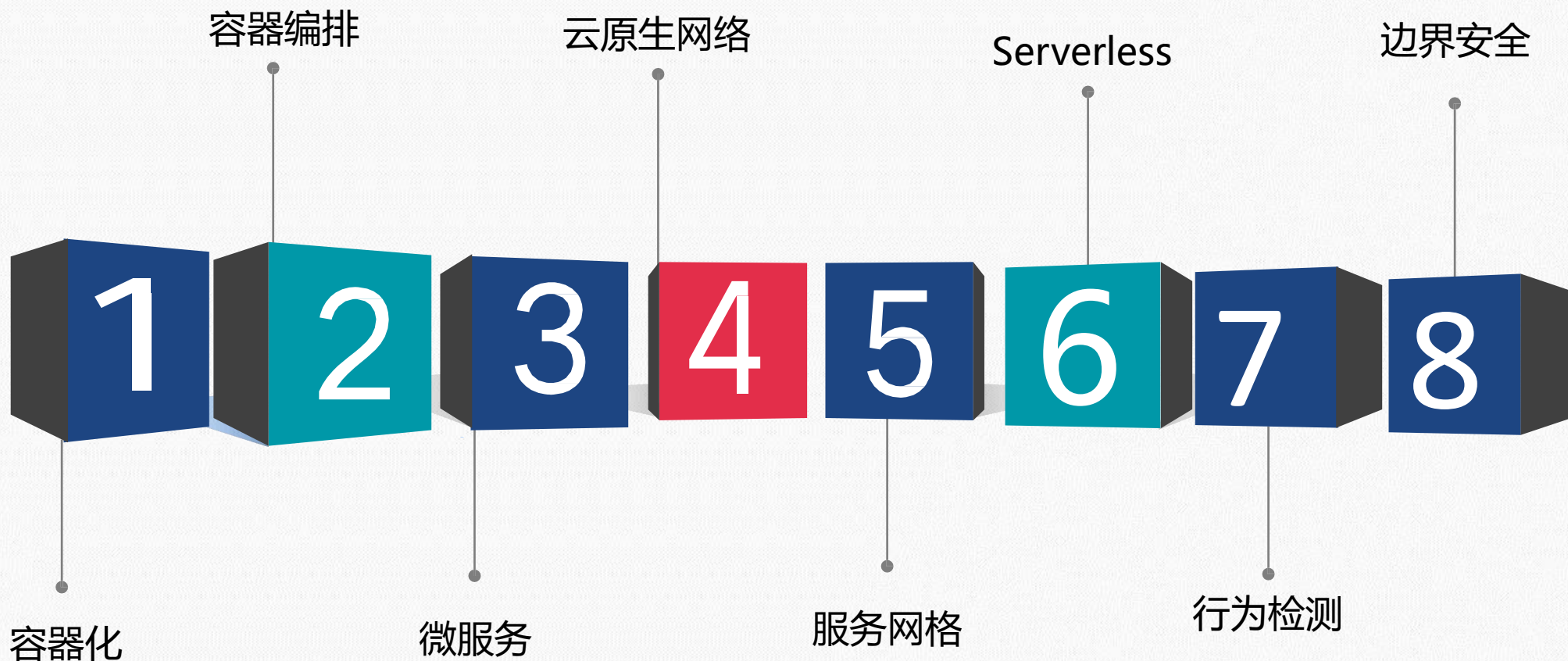
五、云原生场景下开发（dev）面临的挑战



六、云原生场景下运维（ops）面临的挑战



七、云原生环境带来新的安全威胁和风险



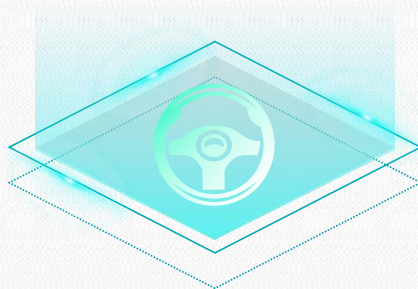
七、云原生环境带来新的安全威胁和风险

1、容器化



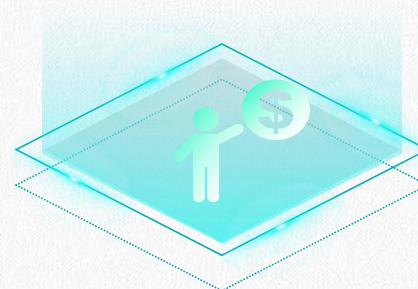
仓库和镜像安全

- 仓库加固或配置策略不足
- 仓库自身漏洞和管理问题
- 镜像获取通道不安全(如明文拉取)
- 镜像含软件漏洞
- 镜像配置缺陷
- 镜像来源不可信



容器逃逸

- 安全容器逃逸风险
- 容器危险配置
- 容器危险挂载
- 相关程序漏洞
- 宿主机内核漏洞



资源耗尽

- 容器内存空间未做限制
- 内存溢出
- 未做好压测
- 拒绝服务



七、云原生环境带来新的安全威胁和风险

2、容器编排



控制权限

不安全配置
编排节点访问控制策略配置不当
非法提权



拒绝服务

编排组件资源使用不设限
不同安全级别容器混合部署
编排工具自身漏洞
基于流量
基于漏洞



七、云原生环境带来新的安全威胁和风险

3、微服务



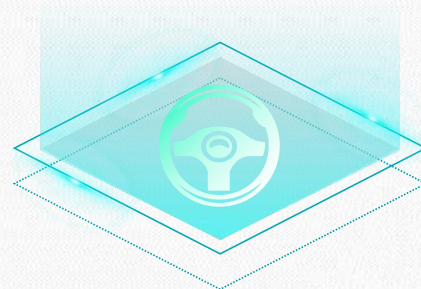
微服务数据泄露

API安全
微服务框架安全



微服务web 应用风险

Owasp top 10
密钥的不规范管理
应用代码安全
未授权访问



微服务间 请求伪造

未授权访问
微服务未做安全防护
应用通信未加密



微服务业务 风险

API滥用
未授权访问



七、云原生环境带来新的安全威胁和风险

4、云原生网络

中间人攻击

8、边界安全

资源暴露面增大

东西流量激增

终端和身份可信状况难把控

7、行为检测

入侵行为检测

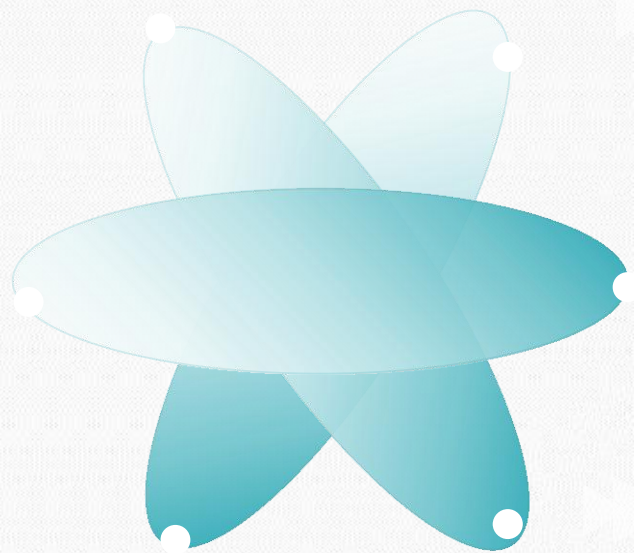
容器逃逸行为

5、服务网格

中间人攻击，越权攻击

6、serverless

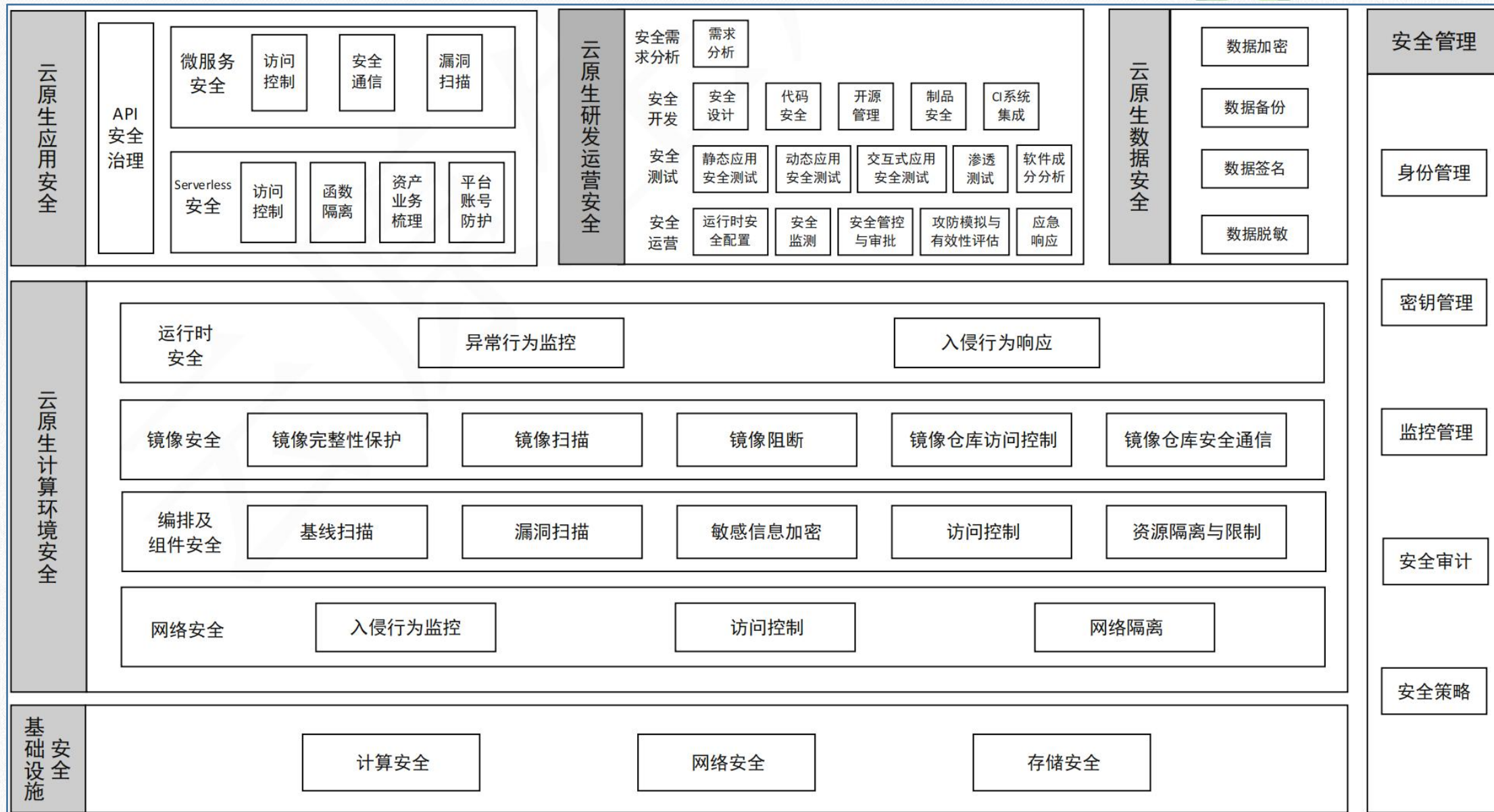
注入、依赖库漏洞、访问控制权限、数据泄露、拒绝服务等



八、云原生场景对安全（sec）技术提出新的要求

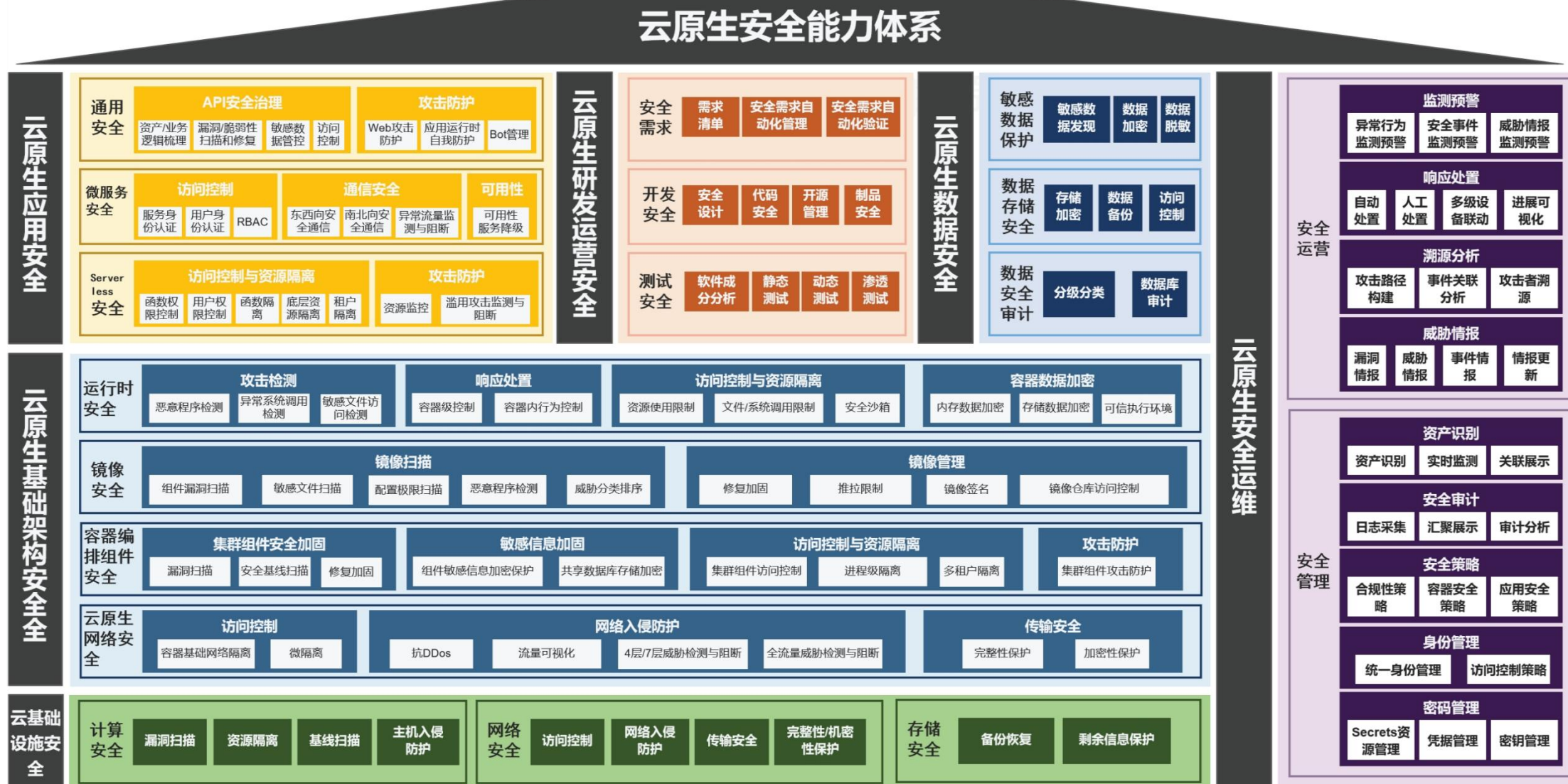


九、云原生安全能力建设：云原生安全防护模型



参考来源：云原生产业联盟 《云原生架构安全白皮书（2021年）》

九、云原生安全能力建设：云原生安全能力体系



参考来源：信通院联合腾讯等多家单位，制定并发布云原生安全能力体系

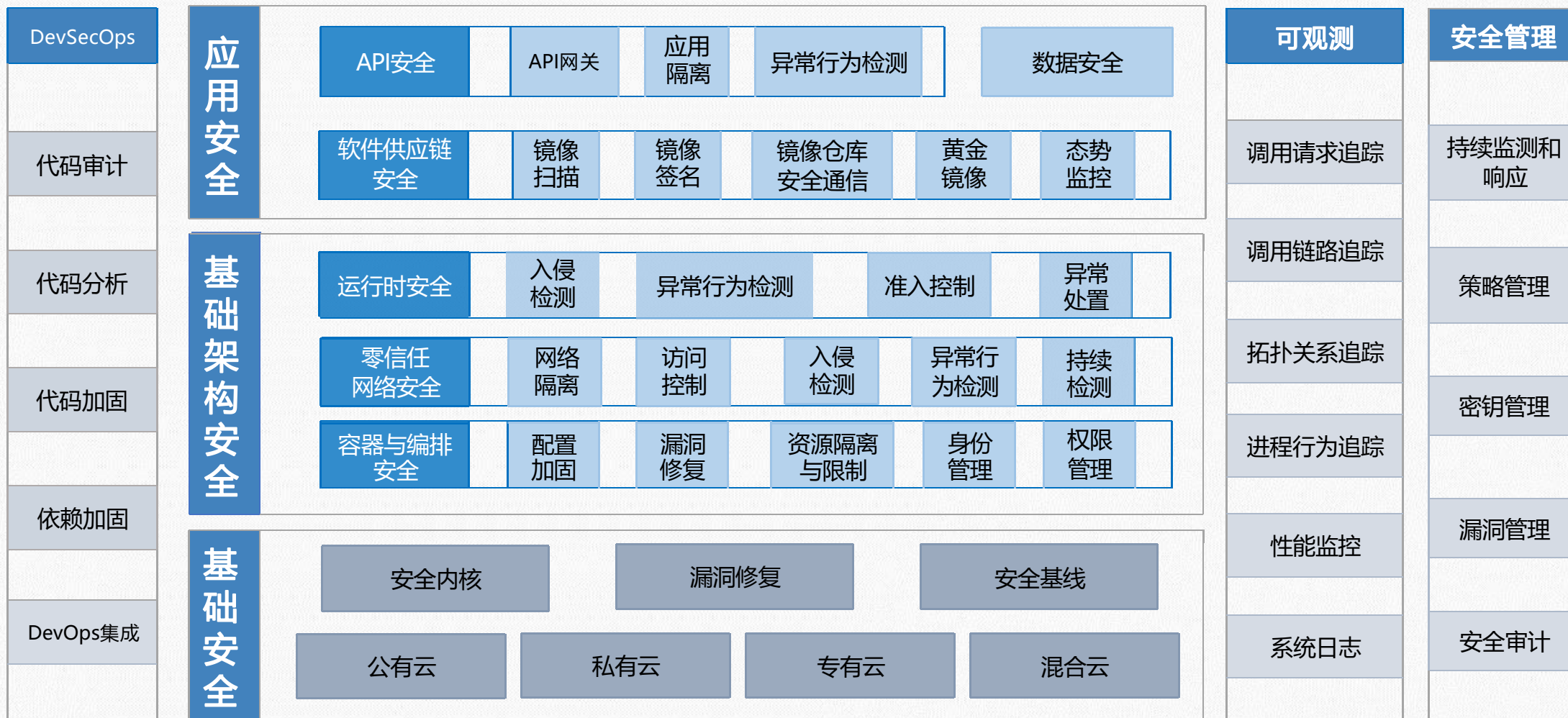


四大设计原则

鉴于云原生技术的特点，以及云原生安全与传统安全建设的区别，在进行容器安全建设时，依托以上几个原则，能更好的指导安全厂商进行容器安全的方案设计和落地实施。



九、云原生安全能力建设：腾讯云容器安全体系框架



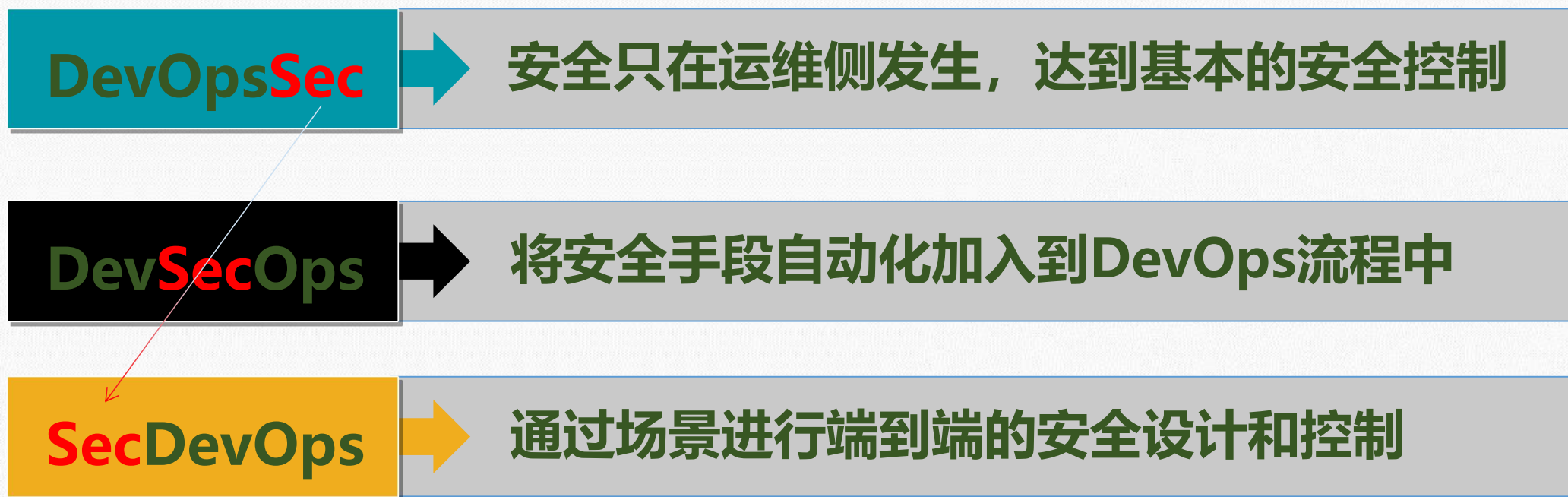


2. DevSecOps如何应对新挑战

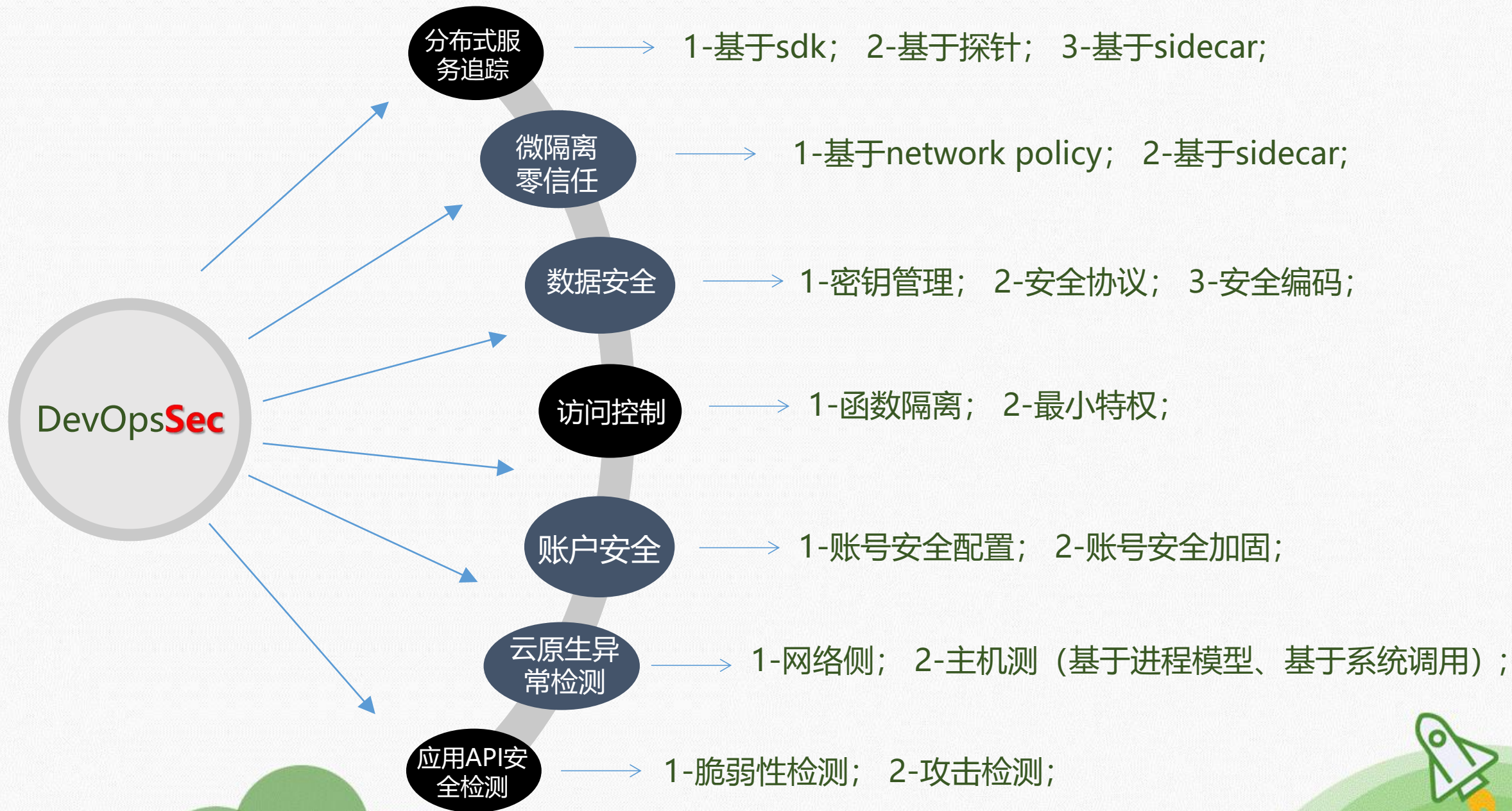
DevSecOps可以理解为将安全性融入到DevOps的过程中，在整个开发和运维的过程中将安全作为一项重要的考虑因素，最终实现应用整个生命周期内的安全性。利用DevSecOps实现安全自动化可以在提高研发运维效率的同时增强应用的安全性。



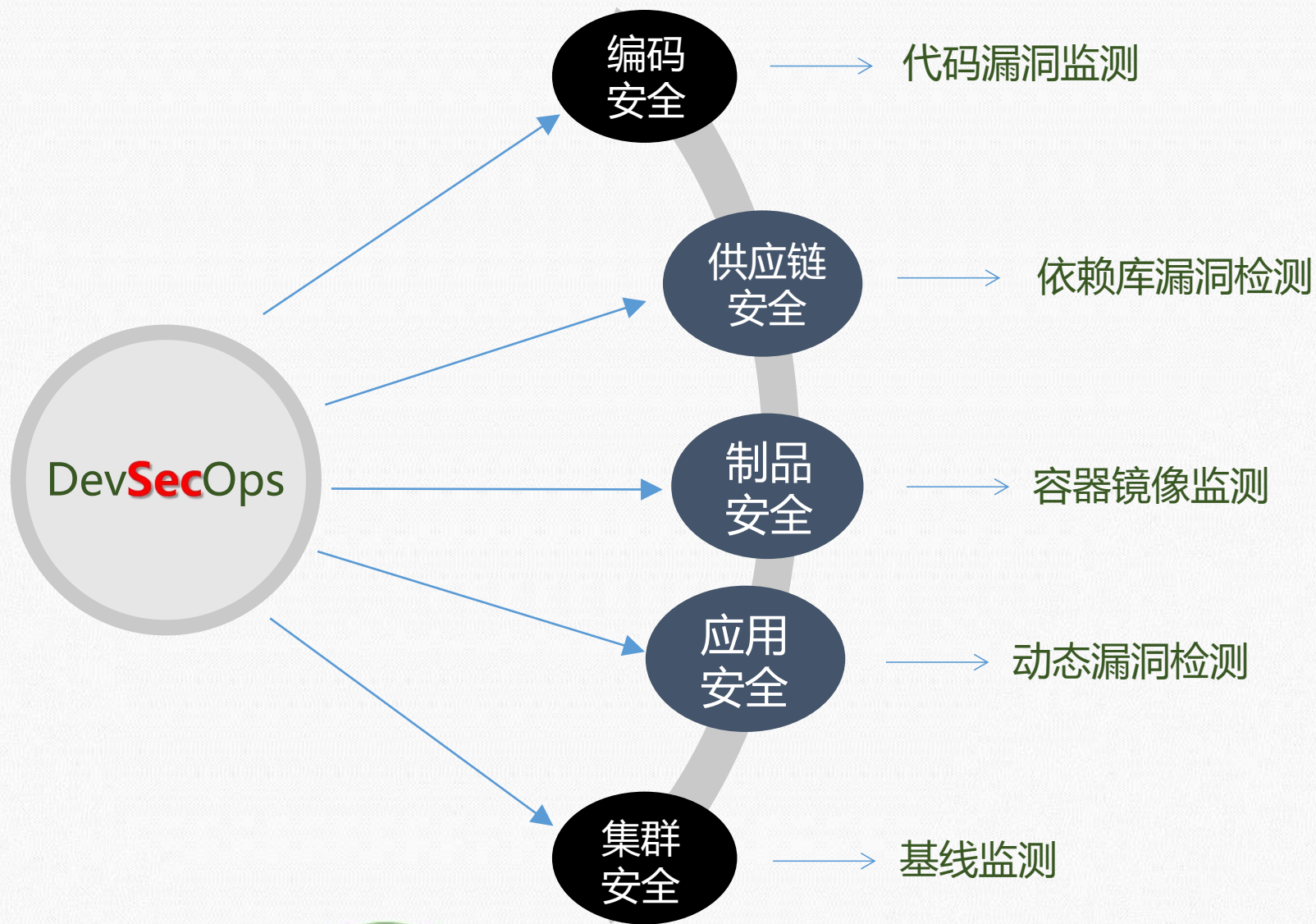
一、大家眼中的DevSecOps = 安全的DevOps ?



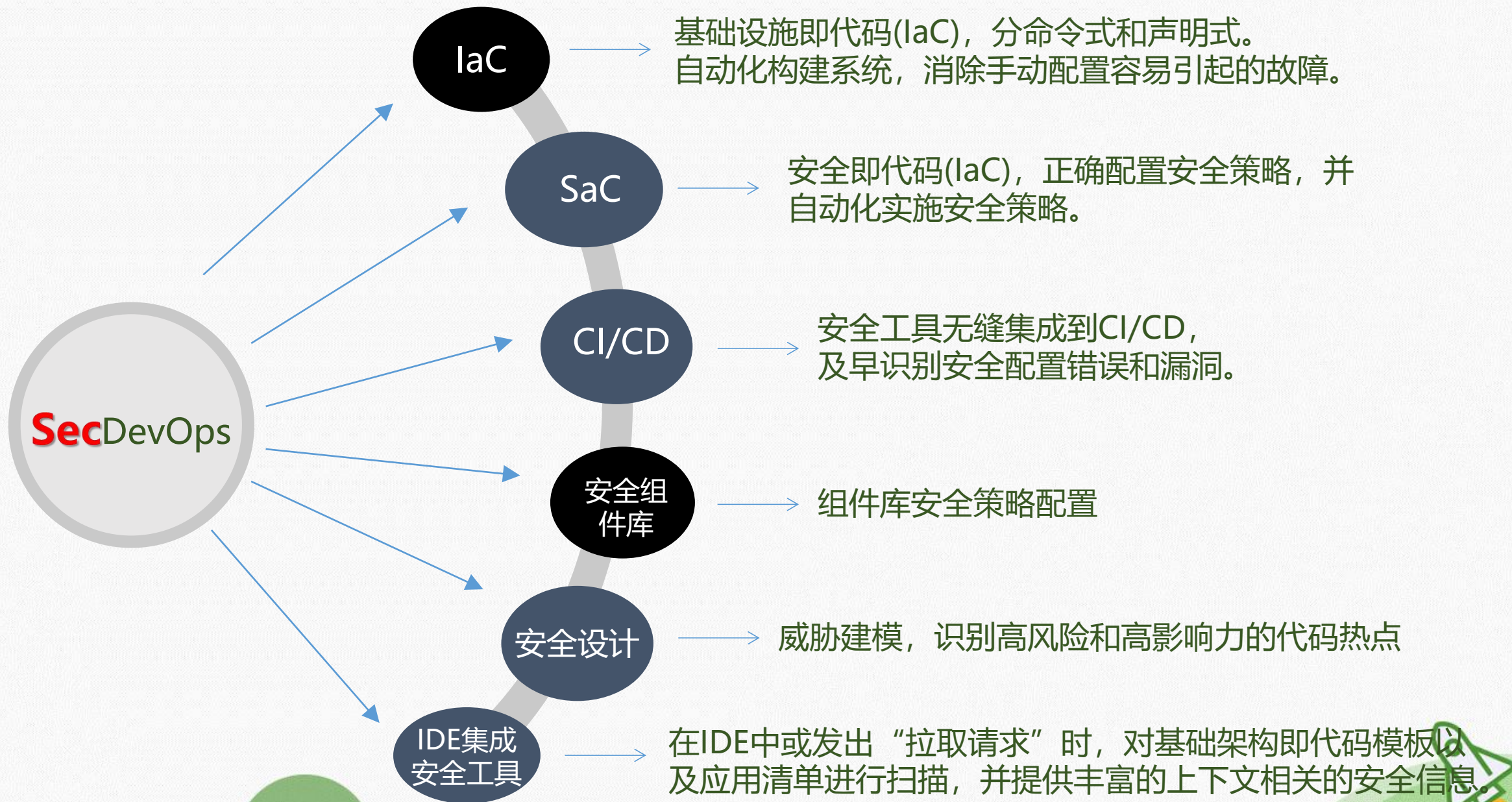
二、DevOpsSec的安全应对措施



三、DevSecOps的安全应对措施



四、SecDevOps的安全应对措施



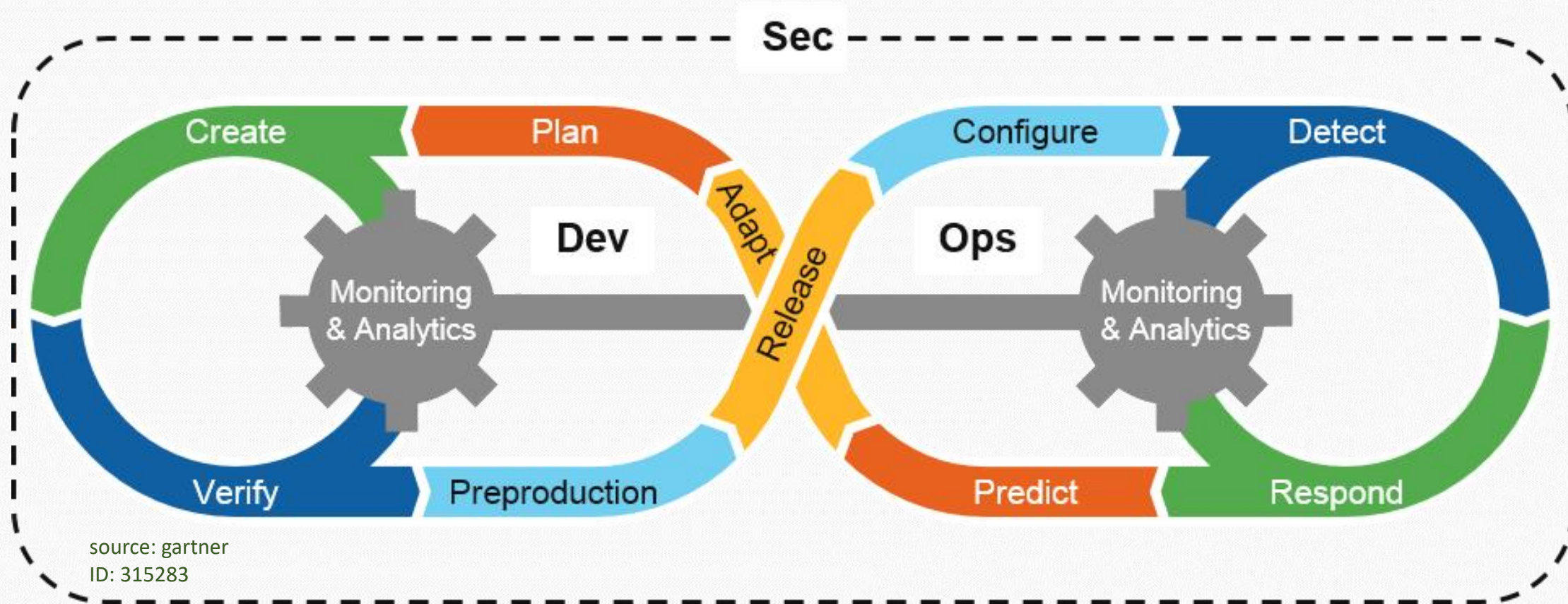


3. 基于免费工具/开源技术打造云原生DevSecOps工具链

在落地DevSecOps过程中，核心的一个部分是安全工具链，通过工具化集成将安全活动内嵌到DevOps研发运营一体化中。



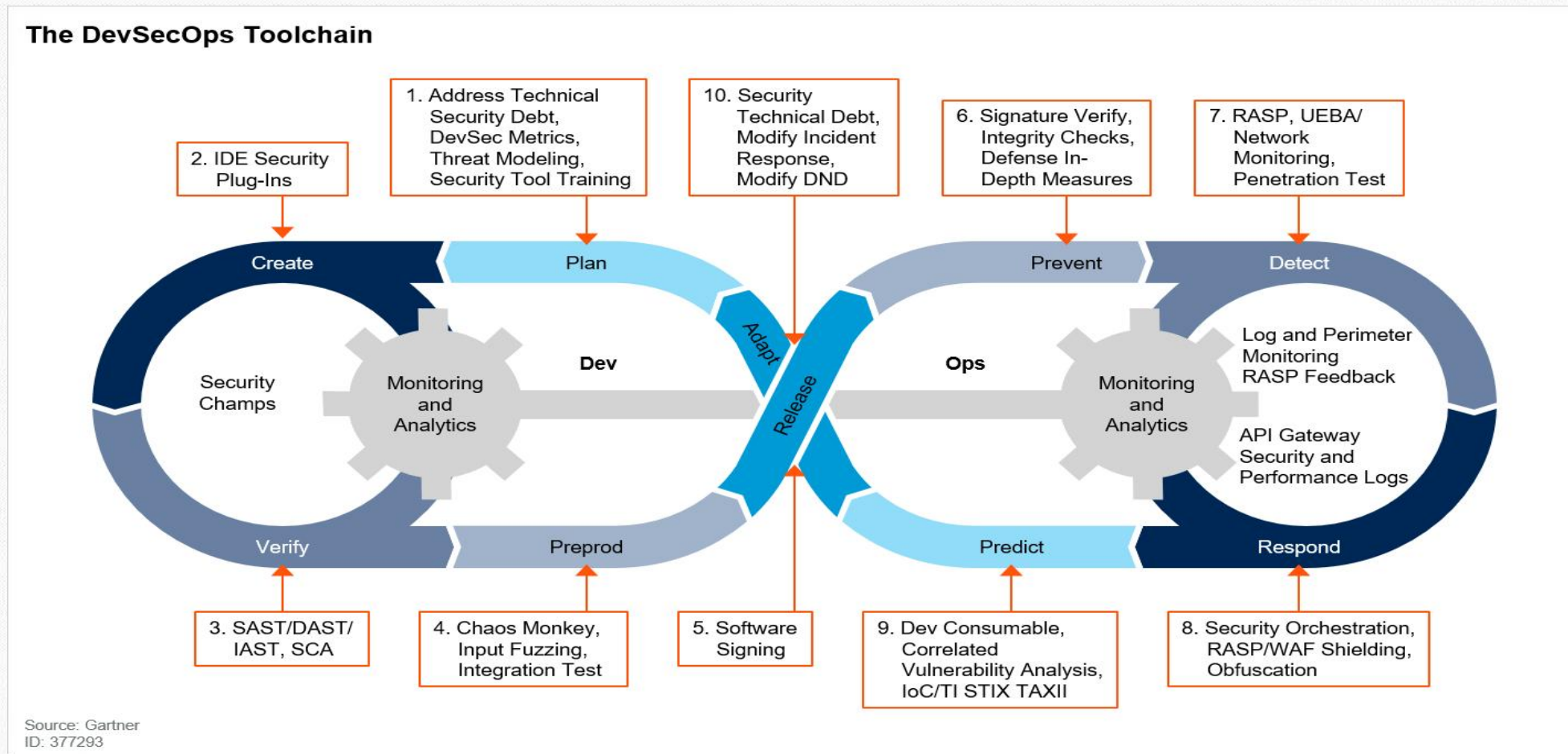
一、DevSecOps模型



2016年，Gartner公开了一份名为“DevSecOps: How to Seamlessly Integrate Security Into DevOps”的研究报告（继2012年gartner提出了DevSecOps的概念），报告详细的阐述了DevSecOps理念和一些实践。一个对DevSecOps最简单直观的图示如上图所示。

DevSecOps可以图形化描绘为，以连续监测和分析为核心，从开发到运行的快速、敏捷迭代。安全的服务交付始于开发，最有效的DevSecOps程序是从开发过程的最初阶段开始，并在其整个生命周期中一直随着流水线推进。

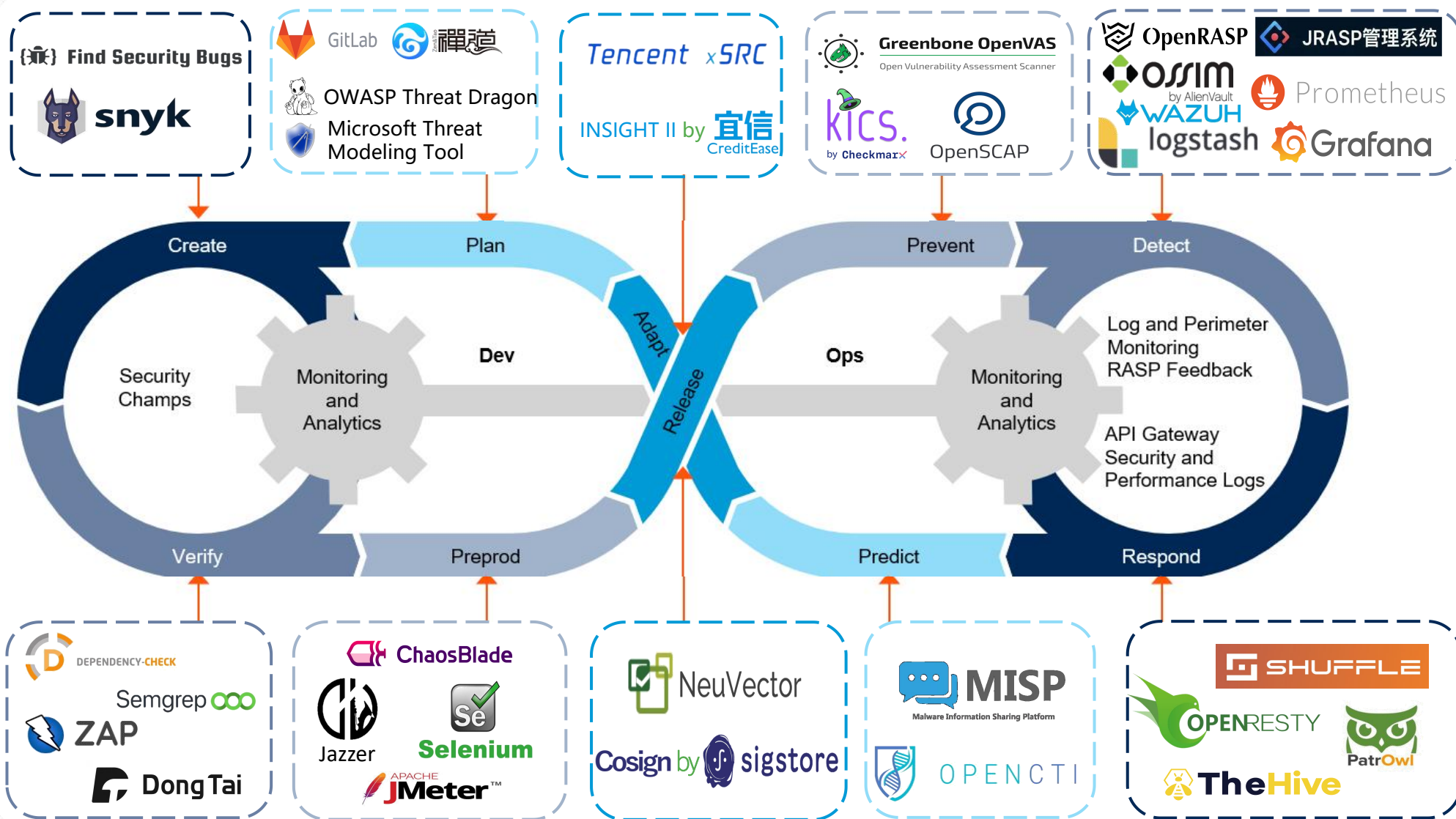
二、DevSecOps工具链实施清单



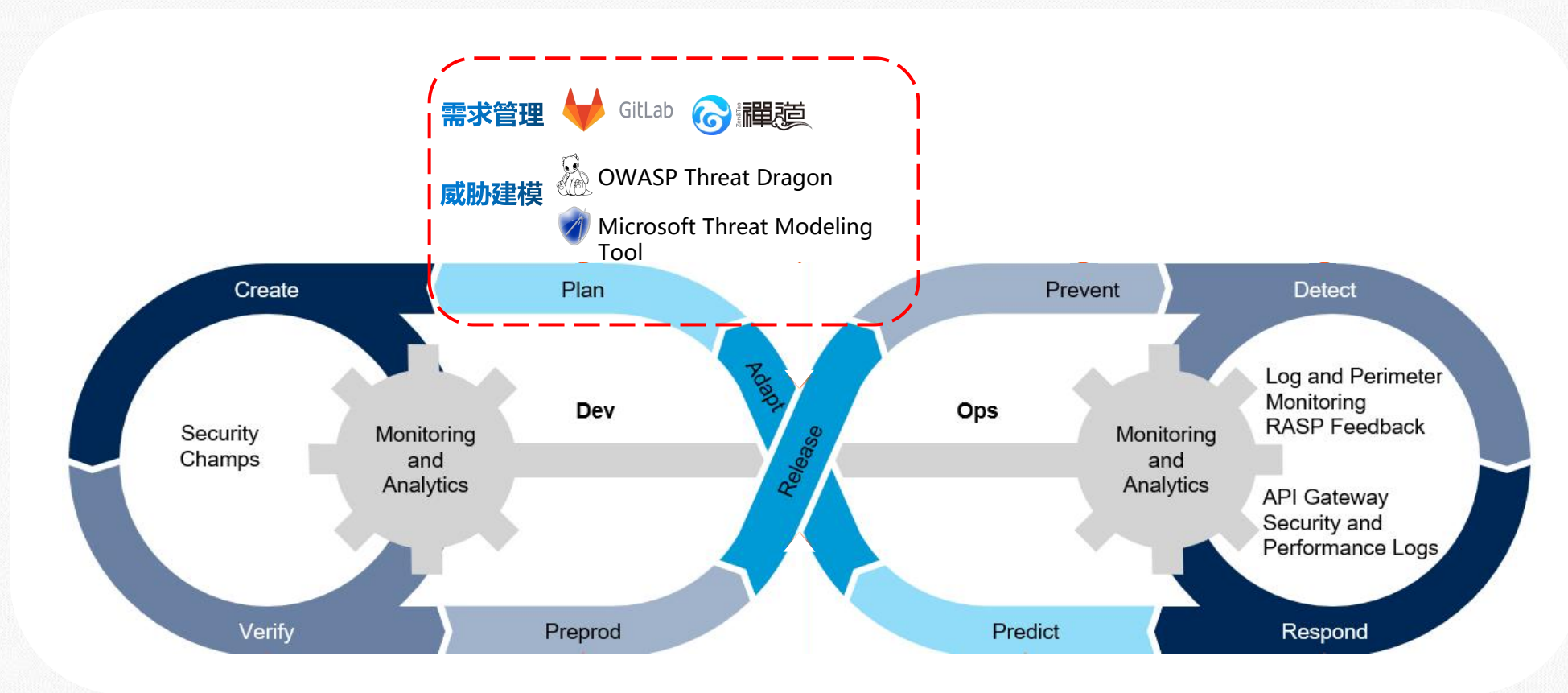
Gartner在2019年的一篇文章中，给出了一个它经过调研和分析之后认为的一个比较全面的实践清单（在DevOps的工具链基础上，增加Sec工具链实践）。

如上图所示，DevSecOps由一系列关键路径和持续的关键步骤中的措施和机制组成，周而复始的运转。它的关注点主要是研发过程中的安全漏洞以及其引发的各类风险的管控。

三、基于免费工具/开源技术打造云原生DevSecOps工具链



3.1、Plan（计划阶段）

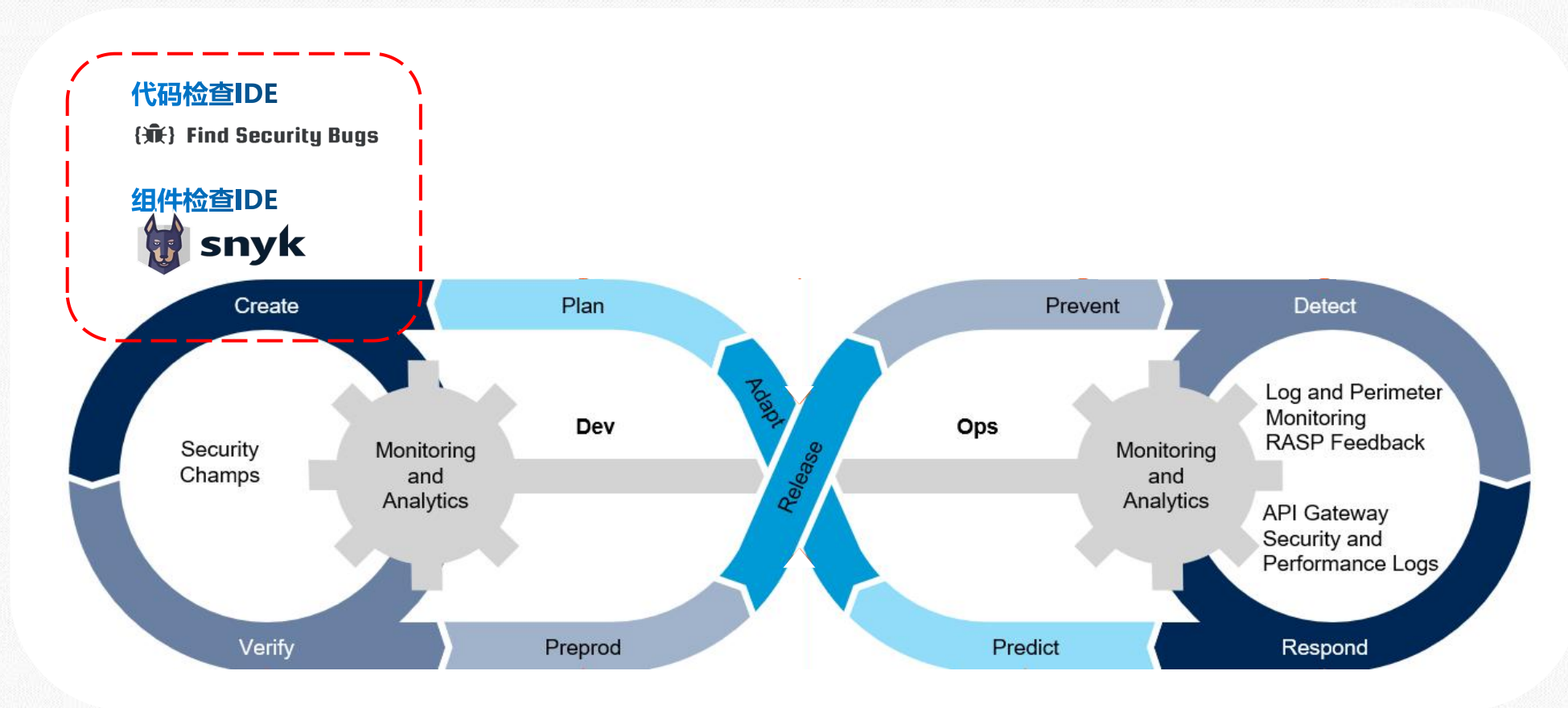


【Plan 需求和设计】

计划阶段是DevSecOps的第一个阶段，其包含了SDL模型里培训、需求、设计等几个阶段，主要关注的是开发前的安全动作。根据Gartner官方工具链模型可以看出其主要是包含有偿还技术安全债务、安全开发衡量指标、威胁建模、安全工具培训。



3.2、Create（创建阶段）

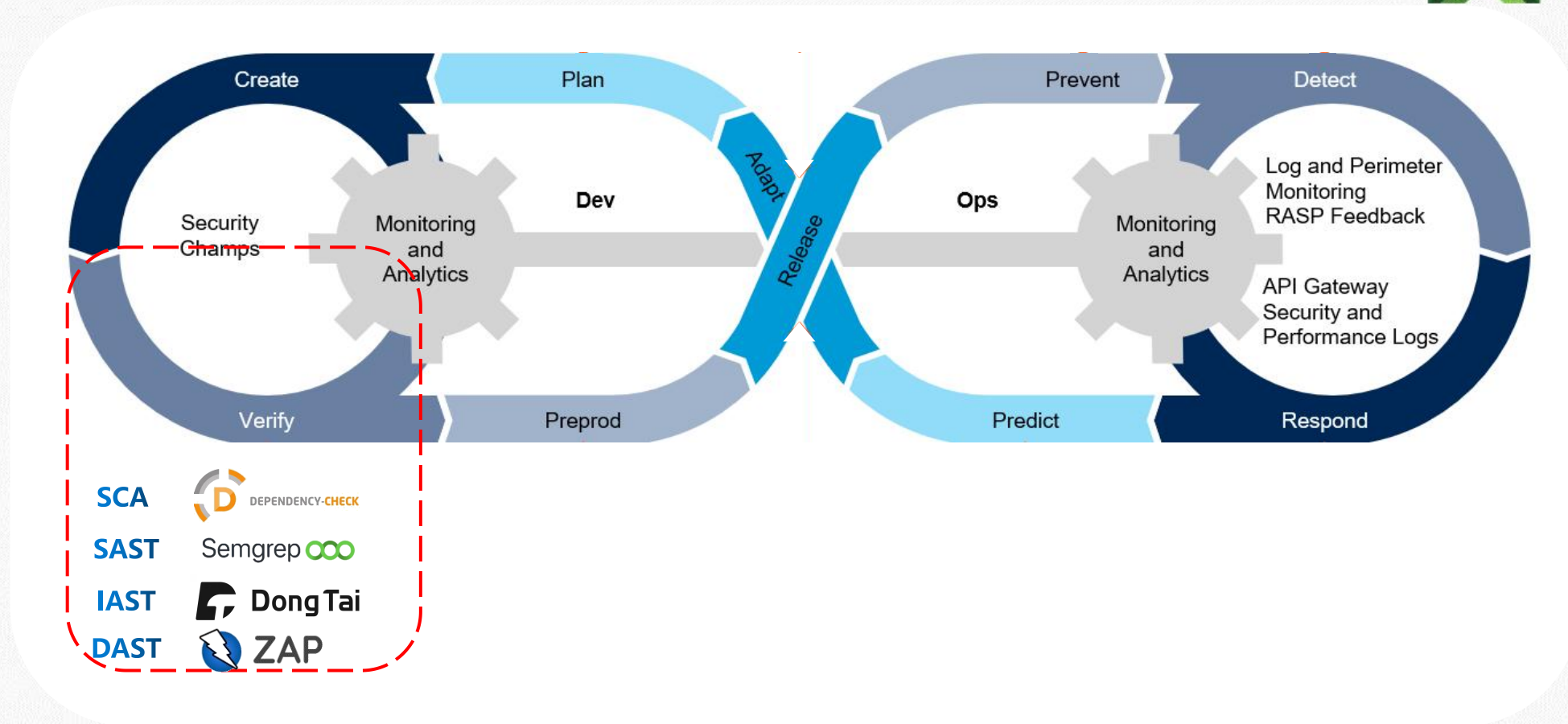


【Create 编码/编译】

创建阶段主要就是指编码阶段。编码阶段主要进行安全编码及检查，旨在在编码阶段进行安全风险的消除。主要包含参考安全编码规范进行安全编码，通过IDE安全插件进行源代码的安全检测，也可以进行安全编码规范的自动化检查，如是否使用不安全或禁用的函数等；甚至是检查代码中是否引用使用了不安全或不合规的第三方组件。



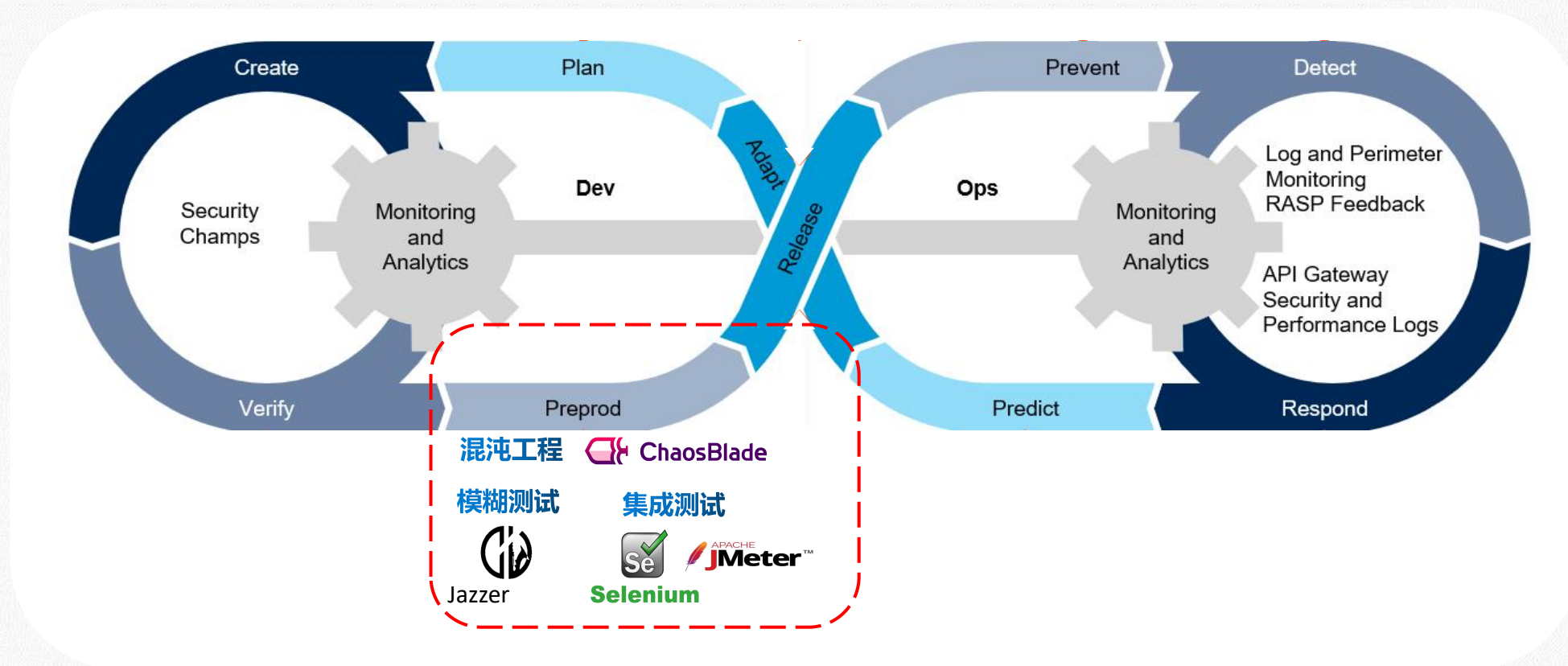
3.3、Verify (验证阶段)



【Verify 测试/验证】

验证阶段其实就是测试阶段，主要以自动化的应用安全测试（AST, Application Security Testing）和软件成分分析（SCA, Software Composition Analysis）为主。应用安全测试主要分为动态应用安全测试（DAST）、静态应用安全测试（SAST）、交互式应用安全测试（IAST,）。软件成分分析主要是关注应用中引入使用的第三方组件的情况。

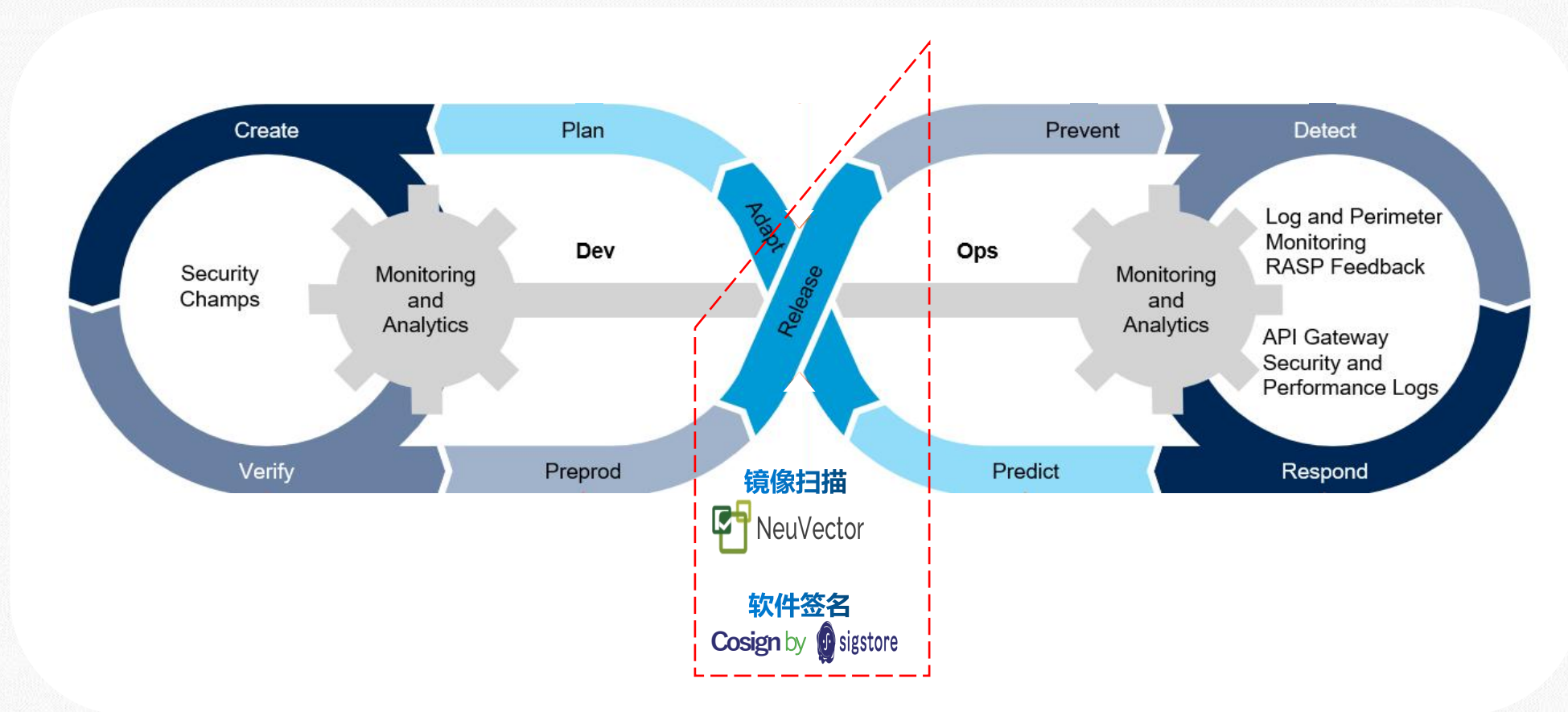
3.4、Preproduction（预发布阶段）



【Preprod 预发布】

预发布阶段一般是测试阶段及正式发布阶段的中间阶段（部分的措施在某些情况下会融合到上一个阶段“Verify”中），其与测试阶段不同的是预发布阶段所发布的预发布环境是连接的正式环境的数据库等，其等同于独立部署的非对外公开的正式环境。在DevSecOps工具链中预发布阶段主要包含有混沌工程（Chaos Engineering）、模糊测试（Fuzzing）、集成测试三个安全动作。

3.5、Release（发布阶段）

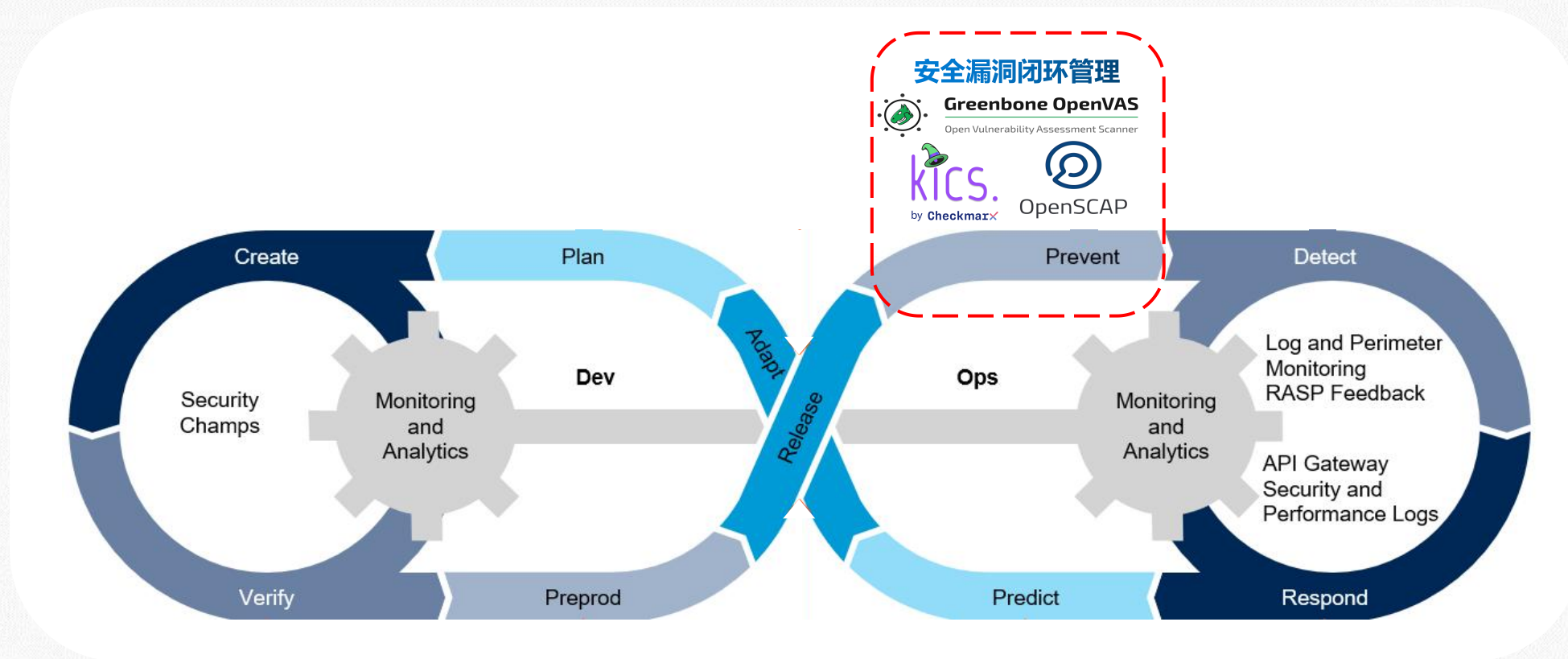


【Release 发布】

针对发布阶段，在DevSecOps的流程中主要动作是软件签名，与预防阶段结合来看，主要是软件防篡改。



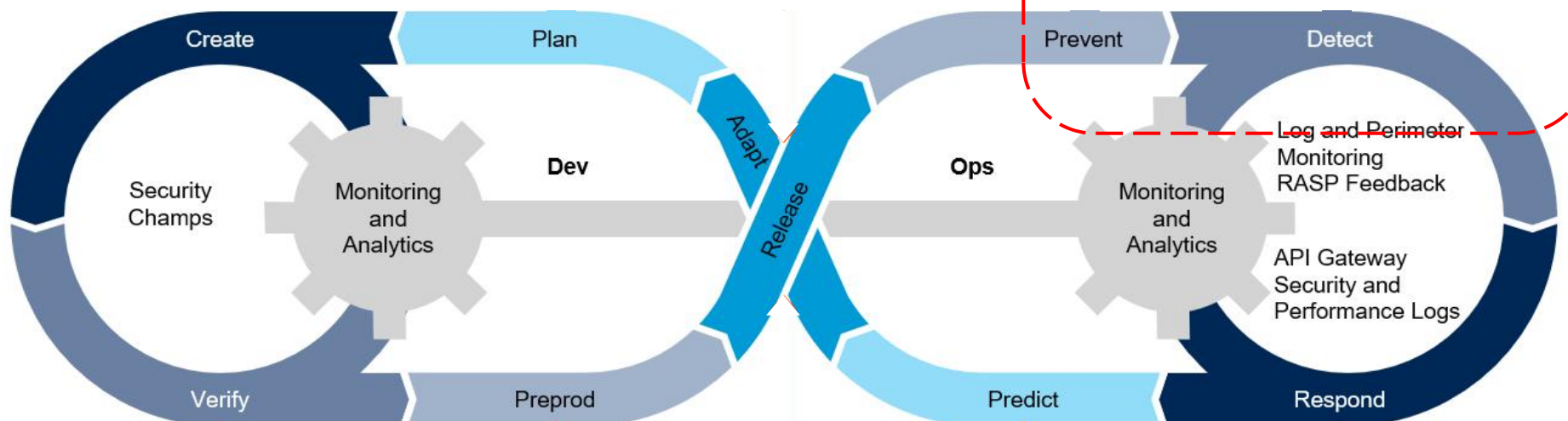
3.6、Prevent (预防阶段)



【Prevent 预防】

预防阶段在早期版本的DevSecOps模型中也被叫为配置 (Configure) 阶段，在最新的模型中被调整为预防 (Prevent)，该阶段理论和实践性还有待进一步发展和细化。该阶段主要包含有**签名验证**、**完整性检查**和**纵深防御**。

3.7、Detect（检测阶段）

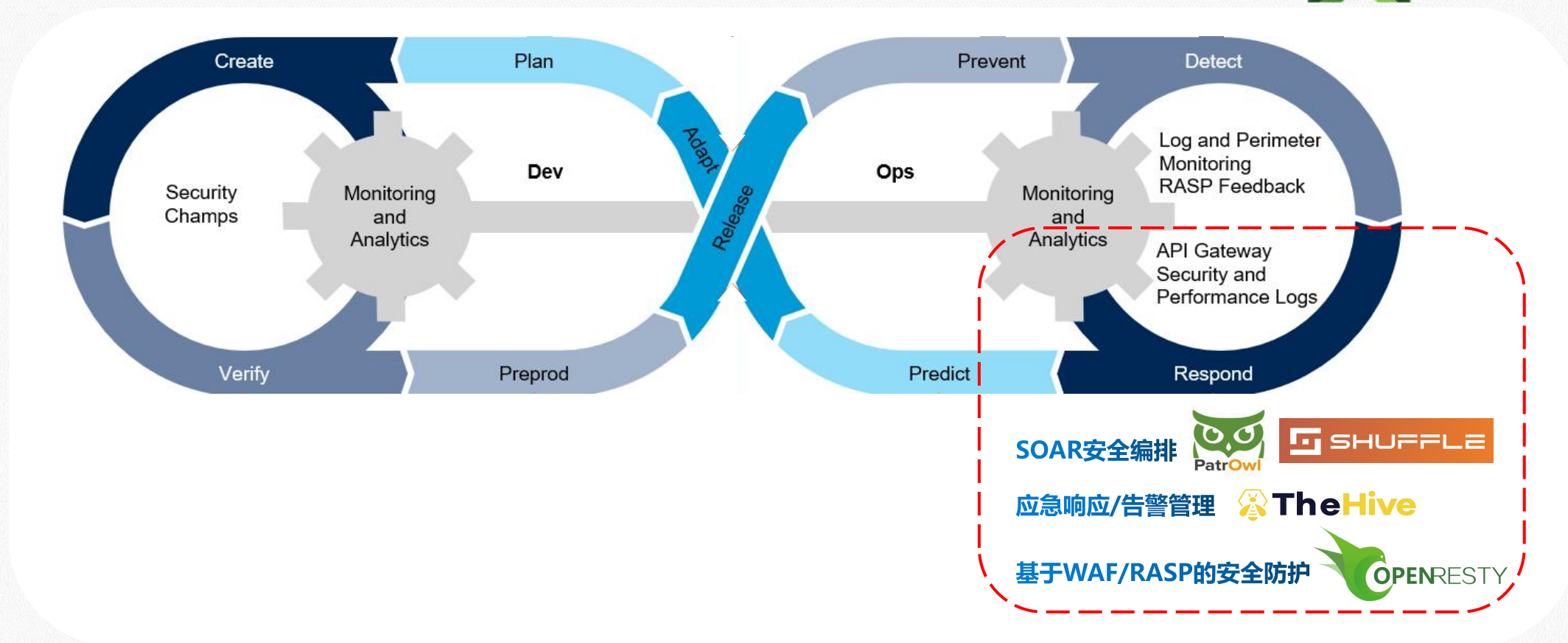


【Detect 检测】

从预防阶段，就已经从开发切换到运维阶段，而检测阶段则更符合传统安全中相关的安全监控动作，该阶段主要包含有**RASP**、**UEBA**、**网络流量监控**、**渗透测试**几个安全动作。



3.8、Respond (响应阶段)

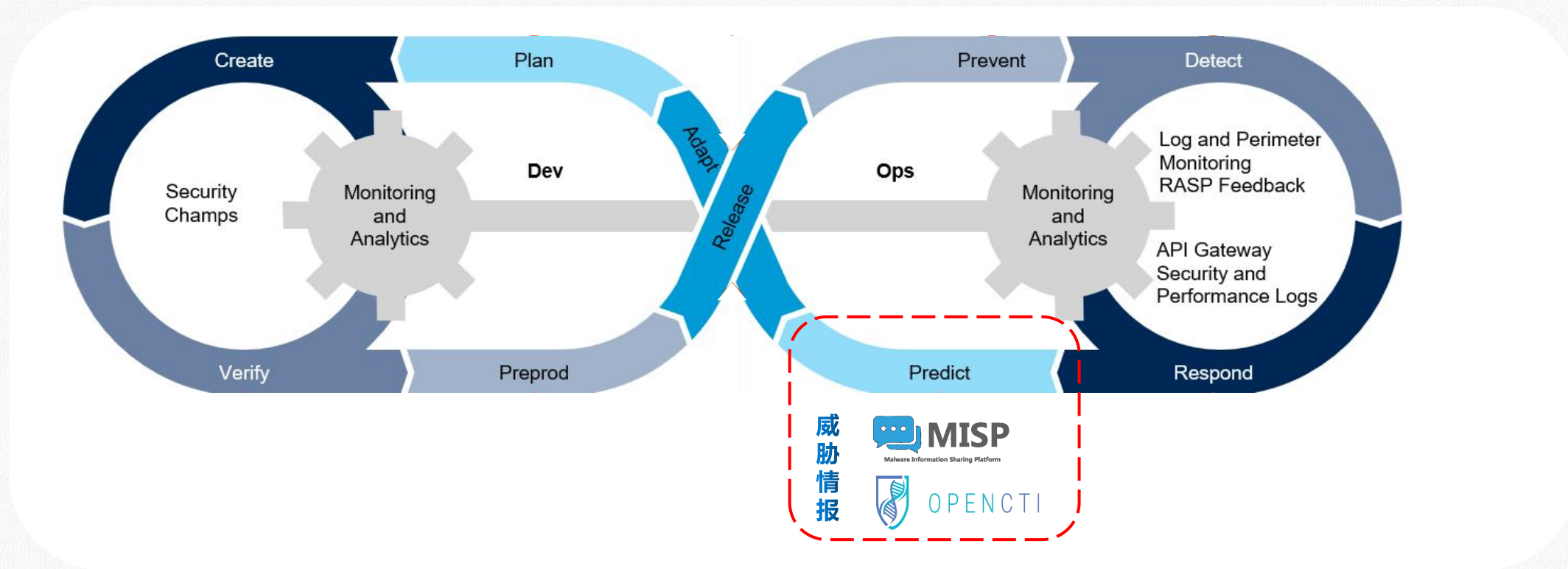


【Respond 响应】

在DevSecOps的响应阶段，安全动作主要包含有**安全编排、基于RASP/WAF的防护、以及混淆**。

基于RASP/WAF安全防护是指通过WAF或RASP的方式，针对 Web 攻击进行拦截阻断，以实现避免由于Web 攻击导致的拖库等后果。而混淆主要是指代码混淆，更多是针对移动APP的混淆操作，避免APP反编译等。安全编排一般是指安全编码自动化和响应(SOAR)，也是Gartner在2017年提出的概念，它是安全编排与自动化 (SOA, Security Orchestration and Automation) 安全事件响应平台 (SIRP, Security Incident Response Platform) 和威胁情报平台 (TIP, Threat Intelligence Platform) 三种技术/工具的融合，目的是快速准确的响应和预测安全事件。

3.9、Predict (预测阶段)



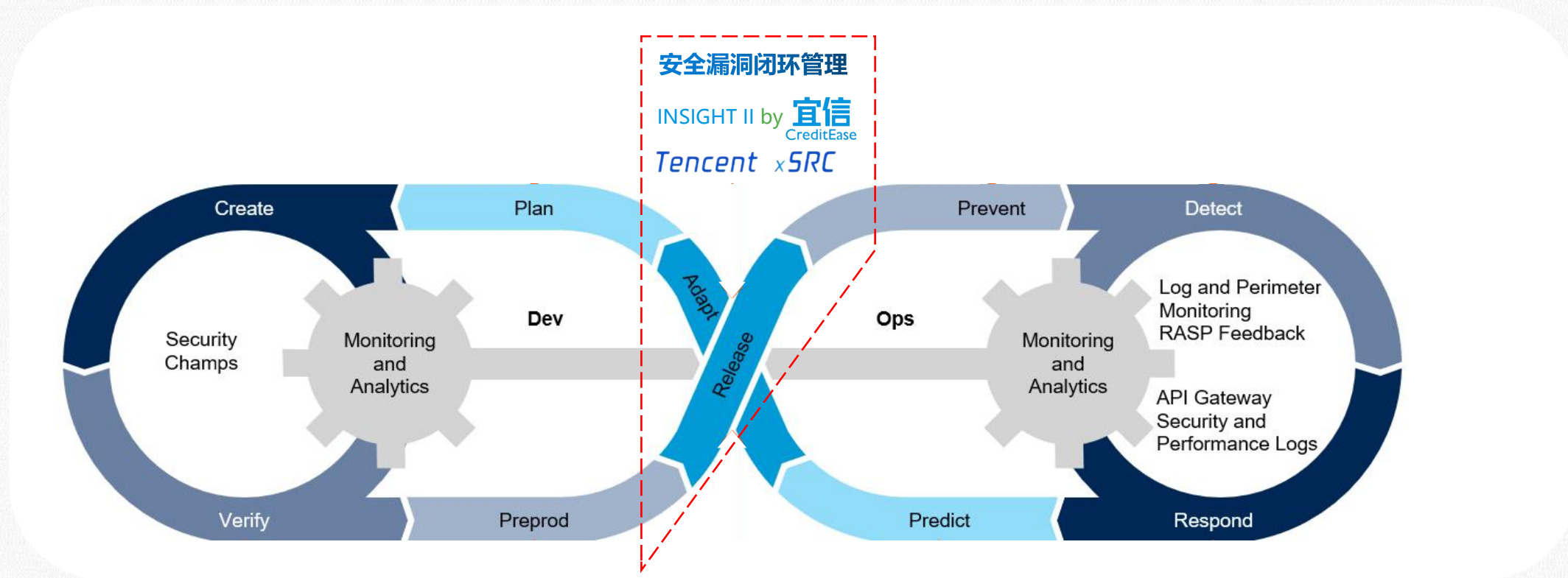
【Predict 预测】

预测阶段主要涉及**漏洞相关性分析与威胁情报**。

漏洞相关性分析 (Application Vulnerability Correlation, AVC) 是一种应用程序安全性工作流和流程管理工具，旨在通过将来自各种安全测试发现的不同数据源的漏洞进行统一管理和自动关联，从而简化漏洞的修复。

威胁情报 (Threat Intelligence, TI) ，是某种基于证据的知识，包括上下文、机制、标示、含义和能够执行的建议，这些知识与资产所面临已有的或酝酿中的威胁或危害相关，可用于资产相关主体对威胁或危害的响应或处理决策提供信息支持。

3.10、Adapt (适应阶段)



【Adapt 适应】

适应阶段主要强调了安全技术债务、修改应急响应方案、安全防御方案等几个点。这个阶段也可以称作优化阶段，主要是基于DevSecOps实施的整个流程的情况，进行持续的适配改进和项目调整优化，对应到过去安全动作，可以理解为持续运营反馈调整的过程，包含对相关安全问题的持续跟踪、闭环，对DevSecOps过程中相关安全动作如策略的调整等。





提问时间

谢谢大家





关注社区公众号
了解更多活动

