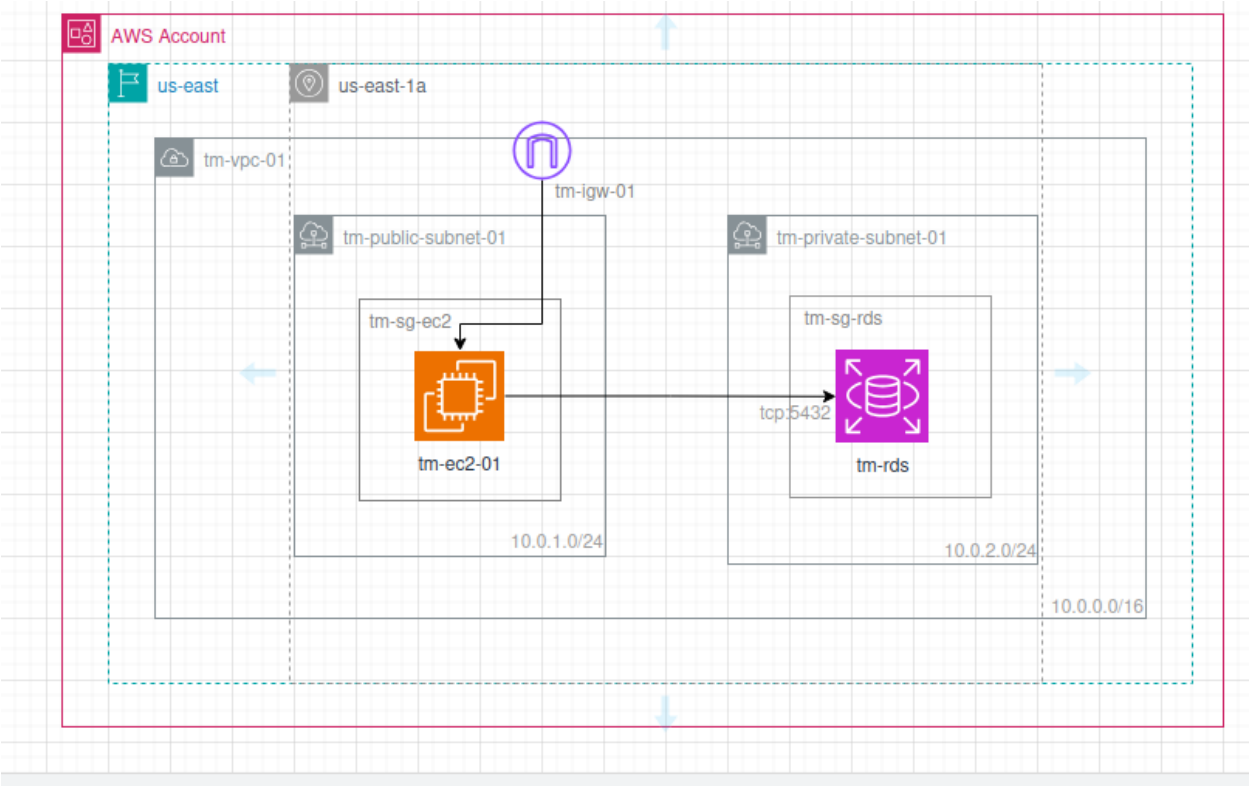
Participante: Andre Svager Allet
Link: [Link diagrama]

Diagrama de arquitetura AWS para solução Toggle Master Monolith



Discussão sobre os 12-Factor App aplicados ao projeto Toggle Master

O 12-Factor App é um conjunto de princípios para construir aplicações cloud-native, escaláveis e fáceis de manter. Quando aplicado em contraste com um MVP monolítico, como é o caso do Toggle Master, onde uma única unidade de deploy entrega funcionalidade de ponta a ponta, temos um impasse. Abaixo vou citar alguns pontos favoráveis, outros contra, e por fim tentar achar pontos que apoiem ou não esta abordagem.

Um monólito pode atender a alguns princípios do 12-Factor, mas surgem tensões conforme o produto cresce. Pontos favoráveis como Time-to-Market Rápido, Infraestrutura mínima, Deploy simples, são ideais para para validação inicial. A questão de baixo overhead cognitivo e operacional, com um único serviço para entender, uma unica pipeline, um unico runtime é ainda ideal para times pequenos e startups iniciais. Como não tem orquestração de serviços e debugging distribuído, torna o Onboarding ainda mais rápido.

Por outro lado, temos a tendência a Estado Oculto, como Sessões em memória, Uso de filesystem local e estado preso à instância, violando processos e disponibilidade, bloqueando a escalabilidade horizontal. Ainda com o modelo fraco de concorrência e escalalabildiade, temos uso ineficiente de recursos, violando o principio da concorrencia. O forte acoplamento a outros serviços traz esquemas de banco fortemente ligados ao código, dificultando trocar ou isolar serviços, violando o principio de Backing Services. Sobre o ponto de infraestrutura, Build / Release / Run Misturados, padrão comum em MVP, gera deploys arriscados, rollbacks difíceis e falta de reprodutibilidade. Por fim, a evolução futura fica comprometida, visto que monolitos são difíceis de quebrar, acumula dívida técnica e força refatorações arriscadas.

A conclusão da decisão, pode se basear na necessidade de cada time ou empresa. Um MVP monolitico pode ser uma decisão acertada quando temos descoberta inicial do produto, domínio ainda pouco claro, time pequeno, baixo tráfego, pressão para entregar rápido. Pode não ser tão boa decisão em cenarios com crescimento de tráfego, crescimento do time, necessidade de escala independente, deploys frequentes, exigências maiores de confiabilidade.