Tucano

why use one computer if u can use 5 with Tucano lol

Eduardo Lemos Paschoalini
Gustavo Dias Aguiar
Luiz Felipe Frazão
Yuri Rousseff

Indíce

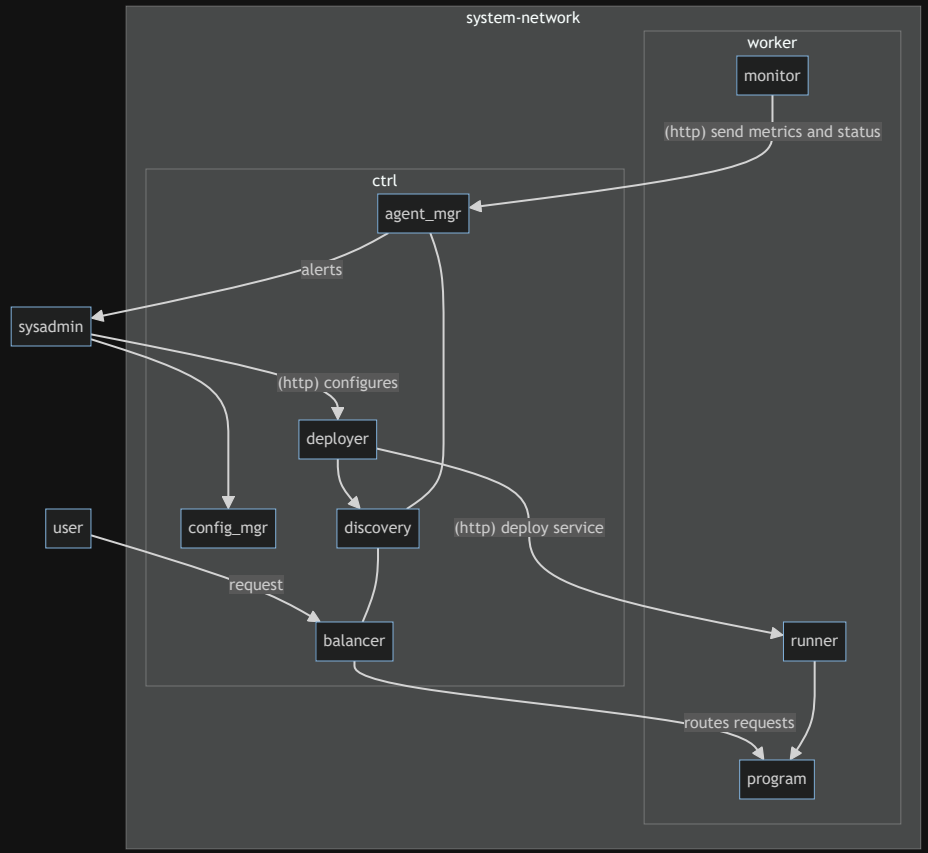
O que é Tucano?

⏰ Simples scheduler de seviços capaz de gerenciar workloads diversos em um sistema composto por vários computadores.

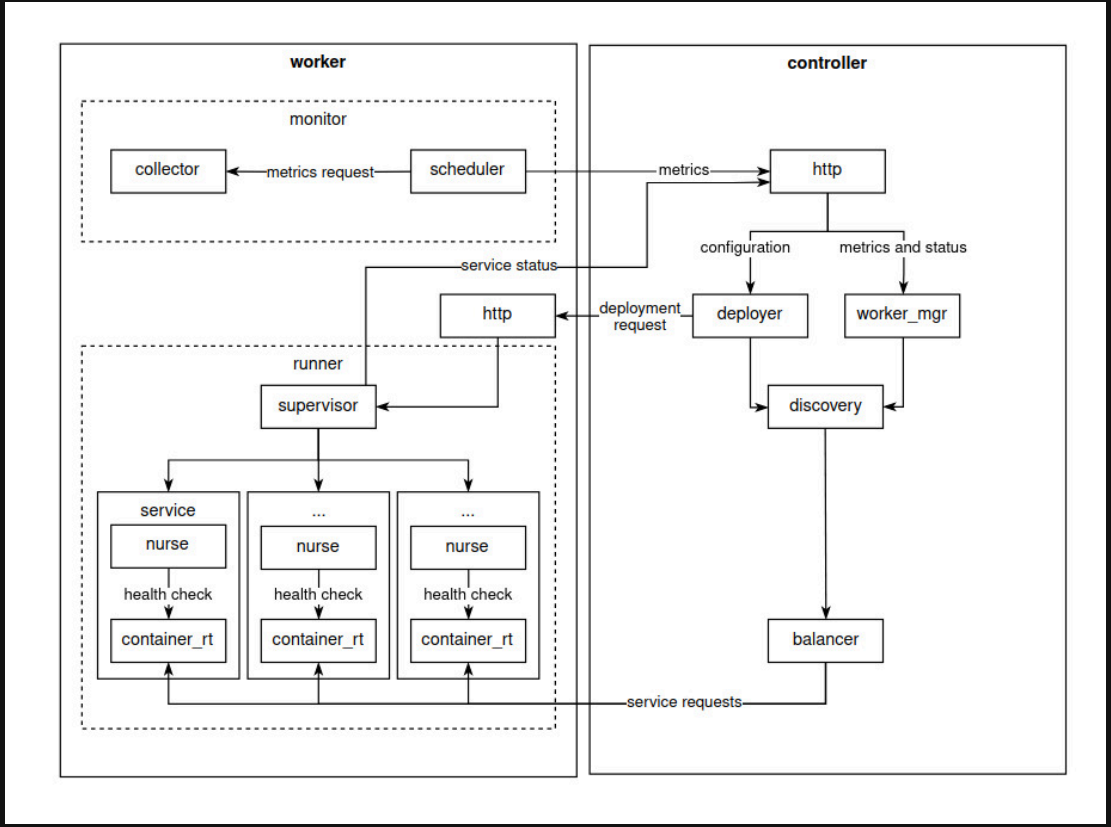
⚖️ Além disso, o scheduler também é responsavel pelo balanceamento de carga, de modo a fazer um service discovery para rotear requisições de usuários ao seus respectivos serviços.

🖳 Uso de multiplos computadores físicos em rede local para uso distribuido

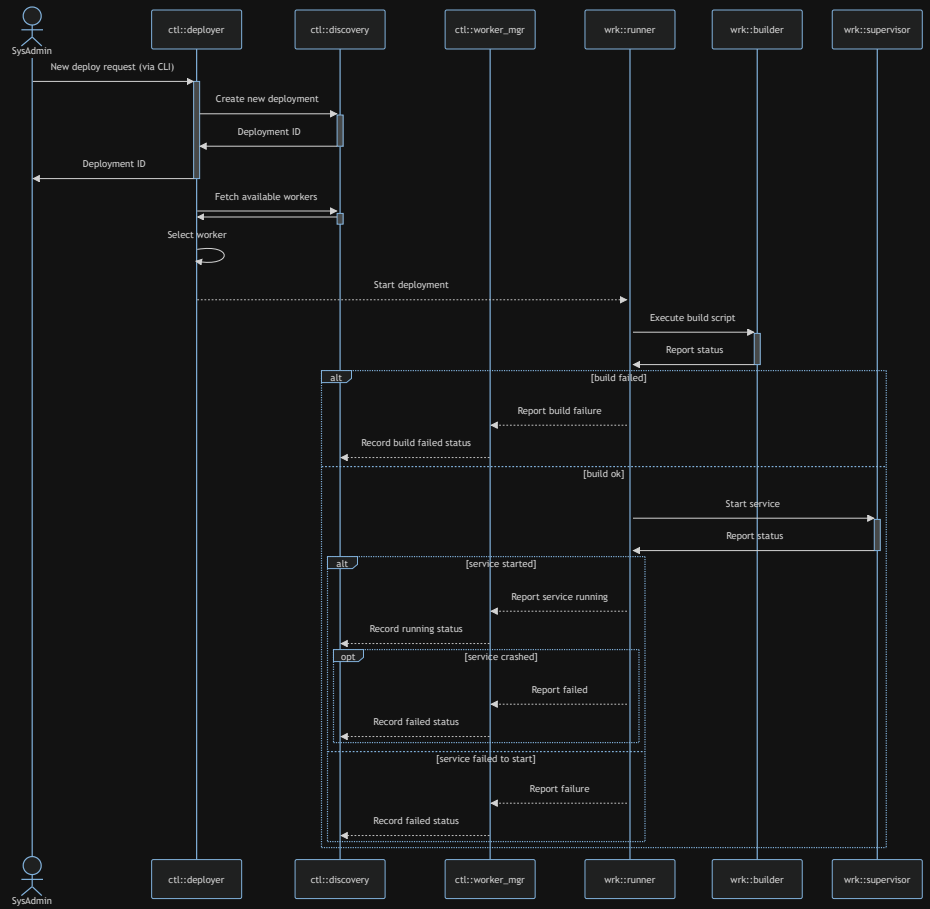
Atualizações



Atualizações



Atualizações



Descrição dos componentes.

Explicação dos componentes dos diagramas.

🎮 Controller

`http*`	Recebe requisições externas dos workers ou admin e roteia para o componente correspondente
`deployer`	Aceita a configuração estática de um serviço e inicia o processo de deploy
`balancer`	Balaceia a carga aos nós correspondentes
`worker_mgr`	Recebe informações dos agents e lida com eventuais "mortes" de workers.
`discovery`	Mantém informações necessárias para realizar service discovery.

Descrição dos componentes.

Explicação dos componentes dos diagramas.

⚒ Worker

`monitor`	Coleta métricas do worker e envia periodicmente ao controlador
`runner`	Recebe instruções de deploy e inicia o processo correspondente no worker

Implementação Controller

HTTP

- O controller usa um servidor HTTP para estabelecer comunicação com o workers e requests de SysAdmin.
- Contém rotas especificas para cada request com seus "handlers" respectivos.

Implementação Controller

Deployer

- Lida com o deploy e redeploy dos serviços. Também trata com a finalização do serviço.

- Parámetros passados em um deploy

```
"name": "Nome do serviço",
"network": {
    "expose_port": 80
},
"scripts": {
    "build-script": "yarn build",
    "runtime-script": "yarn run",
    "teardown_script": "..."
},
"concurrency": 3
```

Implementação Controller

Balancer

- Recebe requests dos usuários e roteia para o serviço correspondente
- O roteamento é descoberto pelo `HOST` da requisição HTTP. O domínio é linkado ao serviço e o `body` e `headers` da requisição original são repassadas ao serviço.
- *Por enquanto, não temos suporte ao protocolo `TLS`*

Implementação Controller

Discovery

- Um dos componentes mais importantes do Controller. Funciona como um database central que mantém informações sobre os workers, serviços, registros de deploys, métricas, etc.

- *Para melhorar a confiabilidade e robustez do sistema mediante a falhas, os dados são persistidos no disco, onde o armazenamento é baseado em um banco de dados SQLite*

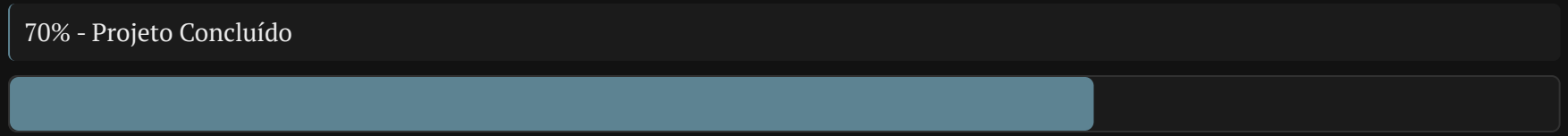
Implementação Worker

Collector

- Coleta informações da máquina worker, como `CPU_Usage` e `Memory`.
- Essas informaçõe são enviadas para o controller, com o objetivo de oferecer insights para balancear a carga.

Progresso do Projeto

- Documentação Concluída
- Diagrama de Deploy
- Componentes do Worker `2/3`
- Componentes do Controller `3/5`



Metodologia

📆 Reuniões semanais

Domingo, 09:00h - 12:00h

Sexta-Feira, 10:40h - 12:20h

Rever progresso, discutir desafios, estabelecer metas

Alinhamento de prioridades e distribuição de tarefas

🔰 Reuniões Ocasionais

Ajustes rápidos, discussões emergenciais

Alinhamento de prioridades e distribuição de tarefas

Sessões de Pair Programming

Obrigado!

GitHub