RUNPOD
사용자가이드

RunPod

RUNPOD
DOCUMENTATION

목차

1. 개요

RunPod는 AI, 머신 러닝 애플리케이션 및 일반 컴퓨팅 요구를 위한 클라우드 컴퓨팅 플랫폼입니다. 우리의 플랫폼을 활용하여 Pod와 서버리스 컴퓨팅 옵션을 통해 GPU 및 CPU 리소스를 사용하여 코드를 실행할 수 있습니다. 오늘 계정을 등록하고 시작해보세요.

Pod란?

Pod는 GPU 및 CPU 인스턴스를 사용하여 컨테이너에서 코드를 실행할 수 있게 해줍니다. Pod는 두 가지 유형이 있습니다.

Secure Cloud와 Community Cloud입니다. Secure Cloud는 T3/T4 데이터 센터에서 운영되어 높은 신뢰성과 보안을 제공합니다. Community Cloud는 개별 컴퓨팅 제공자와 소비자를 검증된 안전한 P2P 시스템으로 연결합니다.

서버리스란?

서버리스는 초 단위로 과금되는 서버리스 컴퓨팅을 제공하며, 프로덕션 환경에서 자동 확장이 가능합니다.

Worker를 정의하고, 이를 위한 REST API 엔드포인트를 생성하고, 작업을 큐에 추가하여 수요에 맞춰 자동 확장을 활용하세요. 이 서비스는 Secure Cloud의 일부로, 낮은 콜드 스타트 시간과 강력한 보안 조치를 제공합니다.

- **시작하기**
  - ✓ 자체 Worker 이미지 빌드하기
  - ✓ vLLM Worker로 LLM 사용하기
  - ✓ Command Line Interface (CLI)

CLI

RunPod는 또한 RunPod 서버리스 플랫폼에서 맞춤형 엔드포인트를 신속하게 개발하고 배포할 수 있는 CLI 도구를 제공합니다.

목표

RunPod는 기능, 사용성, 경험을 저하시키지 않고 누구나 클라우드 컴퓨팅을 접근 가능하고 경제적으로 사용할 수 있도록 지원합니다. 우리는 개인 및 기업이 AI와 클라우드 컴퓨팅의 잠재력을 실현할 수 있도록 최첨단 기술을 제공합니다.

일반 문의사항은 문서를 참조하거나 Discord, 지원 채팅, 이메일을 통해 문의해 주세요. 추가 정보는 연락처 페이지에서 확인할 수 있습니다.

2. 시작하기

RunPod는 인공지능, 머신러닝 애플리케이션 및 광범위한 컴퓨팅 요구를 위한 다용도 클라우드 컴퓨팅 플랫폼입니다.

계정 생성

먼저, 다양한 리소스를 관리하고 접근할 수 있는 RunPod 계정을 생성하세요.
아래 주소를 통해 가입하여 계정을 만드세요. 등록 후 로그인하여 서비스를 시작할 수 있습니다.

https://www.runpod.io/console/signup

계정에 자금 추가

리소스를 배포하기 전에 계정에 자금을 추가해야 합니다. RunPod는 신용카드와 암호화폐 결제를 지원합니다. 여기를 통해 자금을 추가하세요.

https://www.runpod.io/console/user/billing

* 결제 방법 및 청구에 대한 자세한 내용은 청구 정보 페이지나 FAQ에서 확인할 수 있습니다.

리소스 실행

계정에 자금이 추가되면 컴퓨팅 리소스를 배포할 준비가 완료됩니다. 필요에 따라 서버리스 GPU 또는 CPU를 실행하거나 Pod를 임대하여 사용할 수 있습니다.

계정 관리

RunPod를 사용하려면 계정을 생성하거나 팀 멤버로부터 초대를 받아야 합니다.

계정 생성

RunPod.io에서 계정을 생성하세요.

개인 계정을 팀 계정으로 전환

개인 계정은 언제든지 팀 계정으로 전환할 수 있습니다.
1. 콘솔에서 Convert to Team Account를 선택하세요.
2. 팀 이름을 설정하고 전환을 확인합니다.

팀 멤버로 초대받기

팀 멤버가 보낸 링크를 수락하고, Join Team을 선택합니다. 초대하는 방법에 대한 정보는 아래 사용자 초대를 참고하세요.

사용자 초대

RunPod에서 팀 계정이 있는 경우 사용자를 초대할 수 있습니다.

1. Team 페이지로 이동하여 Members 섹션 상단의 Invite New Member 버튼을 선택하세요.
2. 사용자의 역할을 지정합니다.
3. 초대를 생성한 후, Pending Invites 섹션에서 초대 링크를 복사할 수 있습니다.
4. 사용자가 초대 링크를 통해 팀에 가입할 수 있도록 링크를 전송하세요.

역할 유형

RunPod에서 제공하는 역할과 권한은 다음과 같습니다.

- **Basic Role:** 제한된 접근 권한을 가지며 주로 계정 사용 및 기존 Pod 연결만 가능
  - ✓ 계정 사용 및 Pod 연결 가능
  - ✓ 청구 정보 접근 제한
  - ✓ Pod 시작, 중지, 생성 권한 없음

- **Billing Role:** 청구 관리에 특화된 역할
  - ✓ 청구 정보 접근 및 관리 가능
  - ✓ 다른 계정 기능 및 Pod 관리 제한

- **Dev Role:** 개발 작업에 적합한 권한
  - ✓ Basic 역할의 모든 권한 (계정 사용, Pod 연결)
  - ✓ Pod 시작, 중지 및 생성 권한
  - ✓ 청구 정보 접근 제한

- **Admin Role:** 계정에 대한 완전한 제어 권한을 가진 관리자 역할
  - ✓ 모든 계정 기능 및 설정에 대한 접근
  - ✓ 청구 정보 관리 및 접근 가능
  - ✓ Pod 시작, 중지 및 생성 가능
  - ✓ 계정 설정 및 사용자 권한 수정
  - ✓ 모든 계정 리소스와 사용자에 대한 완전한 제어

감사 로그

RunPod에는 수행된 작업을 추적할 수 있는 감사 로그가 포함되어 있습니다. Audit logs 설정으로 이동해 로그를 확인할 수 있습니다.

날짜 범위, 사용자, 리소스, 리소스 ID, 작업별로 감사 로그를 필터링하여 확인할 수 있습니다.

청구 정보

모든 청구는 시간당 컴퓨팅 및 스토리지 요금을 분 단위로 계산합니다.
청구 관련 문의는 Billing FAQ를 참조하세요.

결제 방법

RunPod는 여러 결제 방법을 통해 계정을 충전할 수 있습니다.

- **신용카드** : 신용카드를 사용해 RunPod 계정을 충전할 수 있습니다. 다만, 신용카드 결제가 예상보다 자주 거부될 수 있으며, 그 이유가 명확하지 않을 수 있습니다. 만약 선불 카드를 사용한다면, Stripe의 선불 카드 최소 거래 금액에 따라 예기치 않은 차단을 피하려면 한 번에 $100 이상 충전하는 것이 좋습니다. 자세한 내용은 Stripe가 지원하는 카드 목록을 참조하세요.
- **암호화폐 결제 :** RunPod는 암호화폐 결제를 지원합니다. crypto.com 계정을 미리 설정하고 KYC(본인 확인 절차)를 완료하는 것이 좋습니다. 이는 카드 결제에 문제가 발생할 경우 대체 결제 수단으로 활용할 수 있습니다.
- **비즈니스 청구서 :** $5,000 이상의 대규모 거래의 경우, ACH, 신용카드, Coinbase, 국내 및 국제 송금을 통한 비즈니스 청구서 발행을 제공합니다.

카드 결제에 어려움이 있는 경우, RunPod 지원팀에 문의하여 도움을 받을 수 있습니다.

API 키

API 키는 RunPod에 대한 요청을 인증합니다. 읽기 및 쓰기 권한 또는 읽기 권한을 가진 API 키를 생성할 수 있습니다.

생성

API 키를 생성하려면

1. 콘솔에서 Settings를 선택합니다.
2. API Keys 섹션에서 + API Keys를 선택합니다.
3. 권한을 선택하고 Create를 클릭합니다.

※ **주의**: API 키가 생성되면 이를 안전하게 보관하세요. 비밀번호처럼 다루고, 안전하지 않은 환경에서 공유하지 않도록 주의합니다.

해제

API 키를 삭제하려면

1. 콘솔에서 Settings를 선택합니다.
2. API Keys 섹션에서 휴지통 아이콘을 선택하고 Yes를 클릭합니다.

활성화

API 키를 활성화하려면

1. 콘솔에서 Settings를 선택합니다.
2. API Keys 섹션에서 토글을 선택한 후 Yes를 클릭합니다.

삭제

APIKI를 삭제하려면

1. 콘솔에서 Settings를 선택합니다.
2. API Keys 섹션에서 휴지통 아이콘을 선택한 후 Revoke Key를 클릭합니다.

RunPod에 연결하기

RunPod와 상호작용하는 방법은 여러 가지가 있습니다.

웹 인터페이스

계정을 생성한 후, 다음 주소에서 웹 인터페이스에 로그인하세요. runpod.io/console/signup.

CLI

RunPod의 CLI인 runpodctl을 사용하여 Pods를 관리하고 개발할 수 있습니다.
모든 Pods에는 Pod 범위의 API 키가 설치된 runpodctl이 포함되어 있어, 명령줄을 통해 Pods 관리를 더 쉽게 할 수 있습니다.

SDKs

RunPod는 다음 프로그래밍 언어에 대한 SDK를 제공합니다.

- GraphQL
- JavaScript
- Python

추천 프로그램

RunPod는 사용자가 RunPod 크레딧 형태로 추가 수익을 얻을 수 있는 두 가지 추천 프로그램과 템플릿 프로그램을 제공합니다. 이 문서는 이러한 프로그램에 대한 개요와 참여 방법을 안내합니다.

- **서버리스 추천 프로그램 (BETA)**
  - ✓ 서버리스 추천 프로그램은 추천된 사용자가 서버리스에서 일정 금액을 지출할 때 추천인에게 RunPod 크레딧으로 보상을 제공합니다.
  - ✓ 보상: 추천인은 서버리스 지출의 5%를 RunPod 크레딧으로 적립받습니다.
  - ✓ 자격: 2024년 12월 31일까지 유효합니다.

- **Pod 추천 프로그램 (BETA)**
  - ✓ Pod 추천 프로그램을 통해 사용자는 추천된 사용자가 지출한 금액의 일부를 자신의 계정 기간 동안 받을 수 있습니다.
  - ✓ 보상: 추천인은 GPU Pods에 대해 추천된 사용자가 지출한 금액의 3%를 RunPod 크레딧으로 적립받습니다.
  - ✓ 예시: 만약 20명의 추천인이 각각 GPU Pods에 $100을 지출하면, 추천인은 $60을 적립받습니다.
  - ✓ 자격: 2024년 12월 31일까지 유효합니다.

- **템플릿 추천 프로그램 (BETA)**
  - ✓ 템플릿 프로그램을 통해 사용자는 자신의 Pod 템플릿을 사용한 사용자가 지출한 금액의 일부를 받을 수 있습니다.
  - ✓ 보상: 템플릿 제작자는 자신의 템플릿을 사용하여 지출된 금액의 1%를 RunPod 크레딧으로 적립받습니다.
  - ✓ 예시: 만약 20명의 사용자가 템플릿을 사용하여 시간당 $0.54로 일주일 동안 사용하면, 템플릿 제작자는 $18.14를 적립받습니다.
  - ✓ 자격: 템플릿은 최소 1일의 런타임을 가져야 합니다.

참여 방법

1. 추천 대시보드에 접속합니다.
2. 고유 추천 링크를 찾습니다. 예: https://runpod.io?ref=5t99c9je
3. 추천 링크를 잠재 사용자와 공유합니다.

중요 사항

- 수수료는 구매한 크레딧이 아닌 사용된 크레딧의 비율로 계산됩니다. 따라서 친구가 $1000의 크레딧을 구매하더라도 사용하기 전에는 수수료를 받을 수 없습니다.
- 추천은 추천된 사용자가 귀하의 추천 링크를 사용하여 새 계정을 생성할 때만 유효합니다. 이미 RunPod 계정이 있는 추천인에게는 수수료를 받을 수 없습니다.

지원

추천 프로그램이나 템플릿 프로그램에 대한 질문이나 도움이 필요하시면 RunPod 지원팀에 문의하세요. RunPod는 추천인이 자격 기준을 충족하고 프로그램에 수락되기 전에 발생한 수익을 유지할 수 있도록 허용합니다.

3. Pods

개요

Pods는 실행 중인 컨테이너 인스턴스입니다. Docker Hub, GitHub Container Registry, Amazon Elastic Container Registry 또는 기타 호환 가능한 레지스트리와 같은 컨테이너 레지스트리에서 인스턴스를 가져올 수 있습니다.

> < 참고 >
> Mac(Apple Silicon)에서 RunPod용 이미지를 빌드할 때는 --platform linux/amd64 플래그를 사용하여 이미지가 플랫폼과 호환되도록 해야 합니다. 이 플래그는 RunPod가 현재 linux/amd64 아키텍처만 지원하기 때문에 필요합니다.

Pod 구성 요소 및 설정 이해하기

Pod는 하드웨어에 접근하기 위해 사용자가 생성한 서버 컨테이너로, 동적으로 생성된 식별자를 할당받습니다. 예를 들어, 2s56cp0pof1rmt는 인스턴스를 식별합니다.

Pod는 다음과 같은 여러 구성 요소로 구성됩니다.

- **컨테이너 볼륨**: 운영 체제와 임시 저장소를 포함합니다. 이 저장소는 휘발성이며 Pod가 중단되거나 재부팅될 경우 손실됩니다.
- **디스크 볼륨**: 영구 저장소로, Pod의 임대 기간 동안 보존됩니다. 하드 디스크와 유사하게 이 저장소는 지속적이며 Pod가 중단되거나 재부팅되더라도 사용할 수 있습니다.
- **네트워크 저장소**: 볼륨과 유사하지만 기계 간에 이동할 수 있습니다. 네트워크 저장소를 사용할 경우 Pod를 삭제할 수 있습니다.
- **Ubuntu Linux 컨테이너**: Ubuntu에서 실행 가능한 거의 모든 소프트웨어를 실행할 수 있습니다.
- **할당된 vCPU 및 시스템 RAM**: 컨테이너와 그 내부에서 실행되는 프로세스에 전용으로 제공됩니다.
- **선택적 GPU 또는 CPU**: CUDA 또는 AI/ML 작업과 같은 특정 작업을 위해 맞춤화되어 있지만, 컨테이너 시작 시 필수는 아닙니다.
- **미리 구성된 템플릿**: Pod 생성 시 소프트웨어 및 설정 설치를 자동화하여 다양한 패키지에 간단하게 액세스할 수 있는 원클릭 접근을 제공합니다.
- **웹 액세스를 위한 프록시 연결**: 컨테이너의 모든 열린 포트에 대한 연결을 허용합니다.

< 예시 >

https://[pod-id]-[port number].proxy.runpod.net

또는

https://2s56cp0pof1rmt-7860.proxy.runpod.net/.

시작하기

Pod를 시작하는 방법을 보려면 Pod 선택하기를 참조한 후 Pod 관리 지침을 확인하세요.

더 알아보기

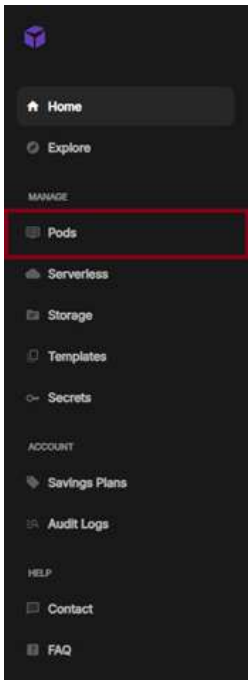
템플릿에서 시작하여 실행 중인 Pod로 바로 이동할 수 있습니다. 더 많은 사용자 지정을 위해 다음을 구성할 수 있습니다.

- GPU 유형 및 수량
- 시스템 디스크 크기
- 시작 명령
- 환경 변수
- HTTP/TCP 포트 노출
- 지속적 저장 옵션

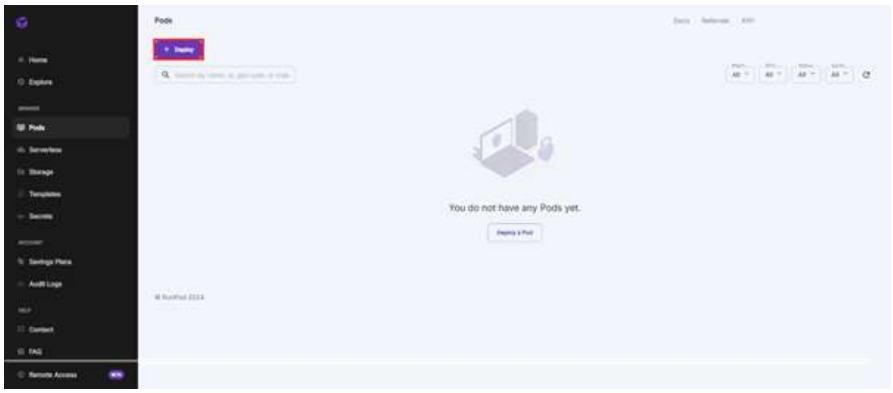
Pod 선택하기와 Pod 관리에 대한 지침을 참조하여 시작하세요.

Pod 생성 간단 요약

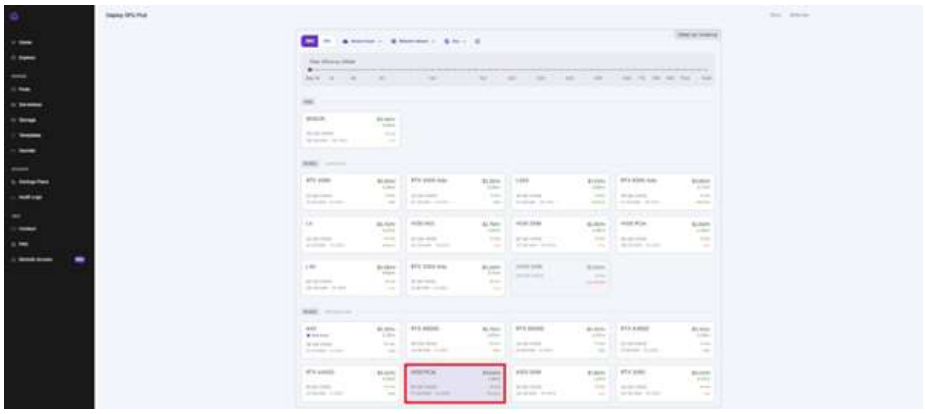
1) 좌측 메뉴에서 Pods 클릭.



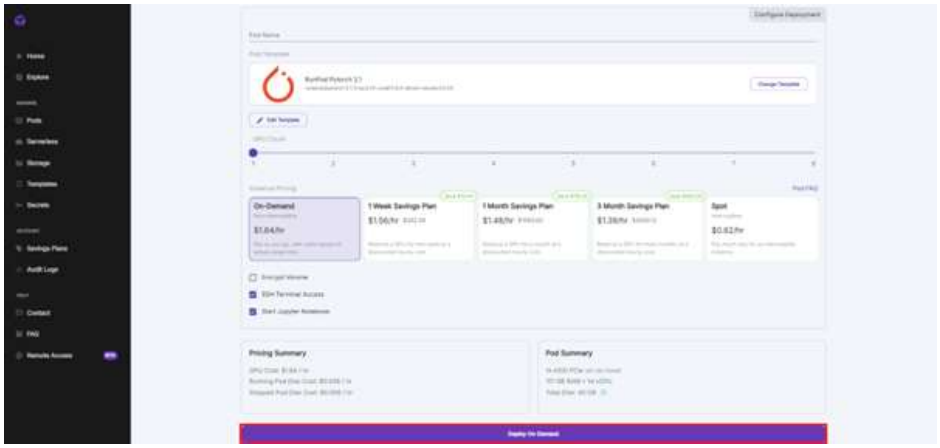
2) 상단에 Deploy 선택.



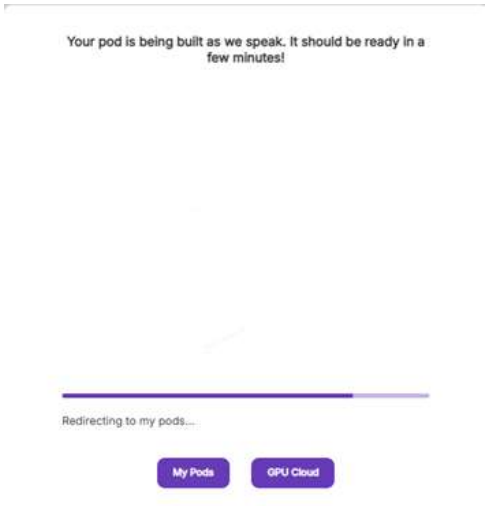
3) 원하는 사양의 GPU 선택.



4) 원하는 사양 선택 후 Deploy 클릭.



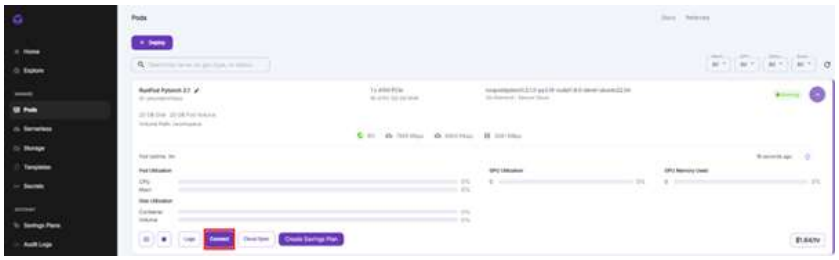
5) 생성 대기.



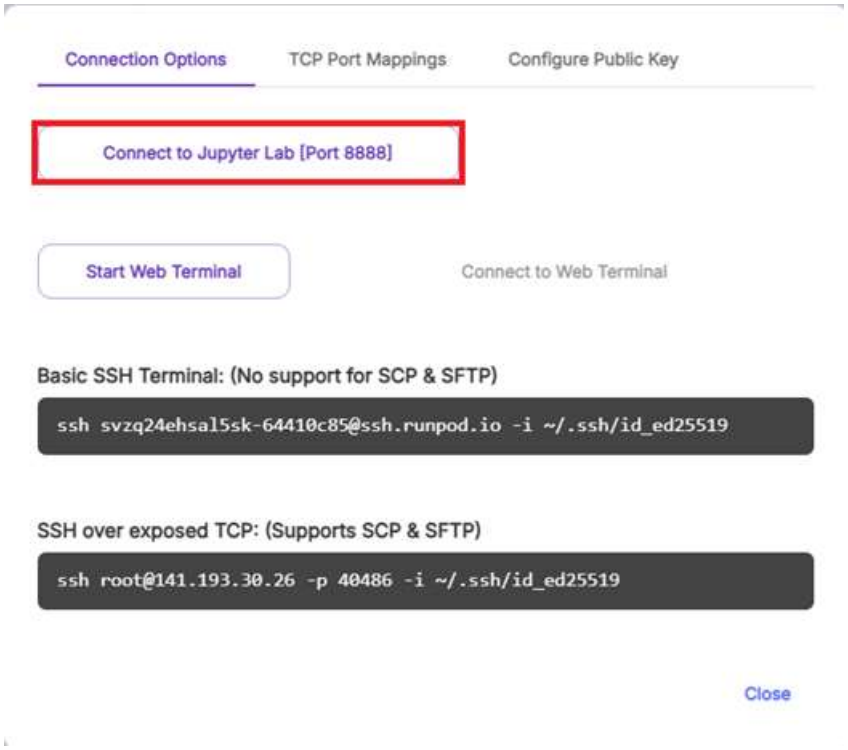
6) 생성 완료 후 Pod에 위와 같이 생성된 Pod 확인.

7) 우측에 화살표 버튼 클릭

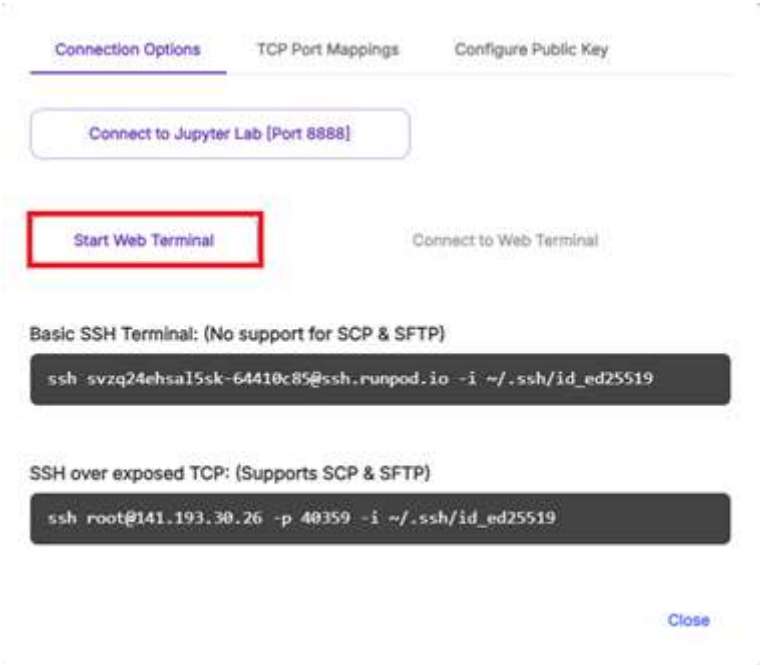
8) Connect 버튼을 클릭하여 접속.



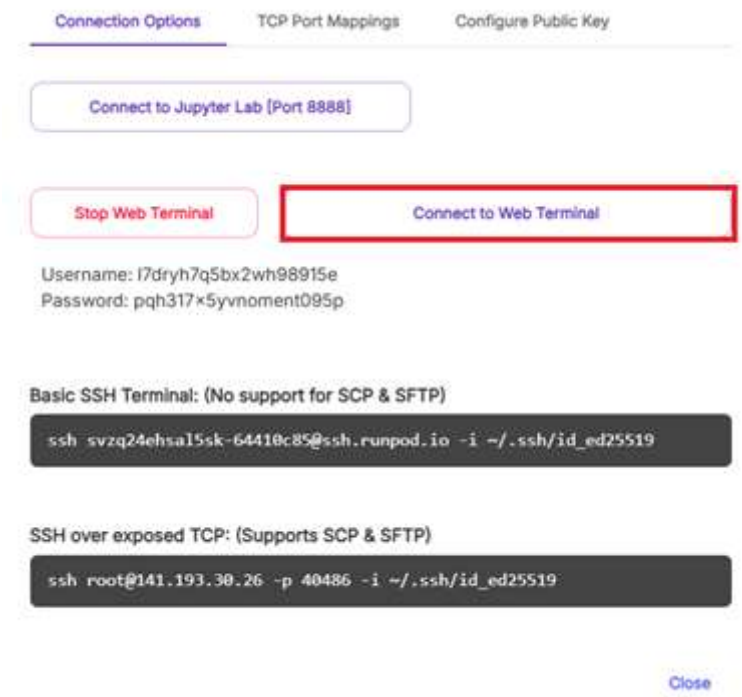
9) Connect to Jupyter Lab을 클릭하여 주피터 노트북 실행



10) Start Web Terminal을 클릭하여 Web 환경에서 터미널 실행.



11) 클릭 후 Connect to Web Terminal 클릭.



12) 접속에 성공 시 위 동작화면 확인

Pod 선택하기

RunPod 배포를 계획할 때 적절한 Pod 인스턴스를 선택하는 것은 매우 중요합니다. VRAM, RAM, vCPU, 임시 및 영구 저장소의 선택은 프로젝트의 성능과 효율성에 큰 영향을 미칠 수 있습니다. 이번 페이지에서는 Pod 구성을 선택하는 방법에 대한 지침을 제공합니다. 하지만 이는 일반적인 가이드라인이므로, 특정 요구 사항을 염두에 두고 계획하는 것이 중요합니다.

개요

모델의 특정 요구 사항을 이해하는 것이 필수적입니다. 일반적으로 Hugging Face와 같은 플랫폼의 모델 카드 설명이나 모델의 config.json 파일에서 자세한 정보를 찾을 수 있습니다.

모델의 특정 요구 사항을 평가하고 계산하는 데 도움이 되는 도구가 있습니다.

- Hugging Face의 모델 메모리 사용량 계산기
- Vokturz의 LLM 실행 가능 계산기
- Alexander Smirnov의 VRAM 추정기

이러한 리소스를 활용하면 Pod에서 어떤 요소를 고려해야 할지 더 명확하게 알 수 있습니다. Pod 선택으로 넘어갈 때 다음 주요 요소에 초점을 맞춰야 합니다.

- GPU
- VRAM
- 디스크 크기

각 구성 요소는 배포의 성능과 효율성에 중요한 역할을 합니다. 이러한 요소와 초기 연구에서

제시된 프로젝트의 특정 요구 사항을 신중하게 고려함으로써, 필요한 Pod 인스턴스를 결정하는 데 잘 준비될 수 있습니다.

GPU

GPU의 유형과 성능은 프로젝트의 처리 능력에 직접적인 영향을 미칩니다. 특히 그래픽 처리 및 머신 러닝과 관련된 작업에서 그렇습니다.

- **중요성**: Pod의 GPU는 복잡한 알고리즘을 처리하는 데 중요한 역할을 합니다. 데이터 과학, 비디오 처리 및 머신 러닝 분야에서 특히 그렇습니다. 더 강력한 GPU는 계산 속도를 크게 높이고 더 복잡한 작업을 가능하게 합니다.

- **선택 기준**

1) 작업 요구 사항: 프로젝트에서 GPU 작업의 강도와 성격을 평가합니다.
2) 호환성: GPU가 사용하려는 소프트웨어 및 프레임워크와 호환되는지 확인합니다.
3) 에너지 효율성: 특히 장기 배포에 대한 GPU의 전력 소비를 고려합니다.

VRAM

VRAM(비디오 RAM)은 무거운 그래픽 처리 및 렌더링 작업에 매우 중요합니다. GPU가 화면에 표시할 이미지 데이터를 저장하는 데 사용되는 전용 메모리입니다.

- **중요성:** VRAM은 집약적인 작업에 필수적입니다. GPU의 메모리 역할을 하여 데이터를 빠르게 저장하고 접근할 수 있도록 합니다. 더 많은 VRAM은 더 큰 텍스처와 더 복잡한 그래픽을 처리할 수 있게 하며, 이는 고해상도 디스플레이 및 고급 3D 렌더링에 중요합니다.

- **선택 기준**

1) 그래픽 강도: 3D 렌더링, 게임 또는 대규모 데이터셋을 포함하는 AI 모델 훈련과 같은 그래픽 집약적인 작업에는 더 많은 VRAM이 필요합니다.
2) 병렬 처리 요구 사항: 여러 데이터 스트림을 동시에 처리해야 하는 작업은 더 많은 VRAM의 혜택을 받습니다.
3) 미래 대비: 더 많은 VRAM을 선택하면 향후 프로젝트 요구 사항에 더 적응할 수 있습니다.

Storage

임시 및 영구 저장소 모두에 충분한 저장소를 확보하는 것은 원활한 작동 및 데이터 관리를 보장합니다.

- **중요성:** 디스크 크기(임시 및 영구 저장소 포함)는 데이터 저장, 캐싱 및 프로젝트 운영에 필요한 공간을 보장하는 데 매우 중요합니다.

- **선택 기준**

1) 데이터량: 프로젝트에서 생성하고 처리할 데이터의 양을 추정합니다.
2) 속도 요구 사항: 더 빠른 디스크 속도는 전체 시스템 성능을 향상시킬 수 있습니다.
3) 데이터 유지 필요성: 데이터 유지 정책에 따라 임시(휘발성) 저장소와 영구(비휘발성) 저장소 간의 균형을 결정합니다.

이 정보를 바탕으로 Pod 인스턴스를 신중하게 선택하여 최적의 성능을 이끌어내세요.

Pods 관리하기

RunPod에서 Pods를 시작하고 중지하며 관리하는 방법에 대해 알아보세요. 여기에는 Pods 생성 및 종료, 명령줄 인터페이스를 사용한 Pods 관리가 포함됩니다.

- **필수 조건 :** RunPod CLI를 사용하는 경우, 구성에서 API 키를 설정해야 합니다。

```
runpodctl config --apiKey $RUNPOD_API_KEY
```

여기서 $RUNPOD_API_KEY를 자신의 RunPod API 키로 대체하세요.

API 키가 설정되면 인프라를 관리할 수 있습니다. 어떤 Pod가 필요한지 확실하지 않은 경우, Pod 선택하기를 참조하세요.

Pods 생성하기

- **웹**

1. Pods로 이동하여 + Deploy를 선택합니다.
2. GPU와 CPU 중 선택합니다.
3. 다음과 같은 설정으로 인스턴스를 사용자 정의합니다:
   - ✓ (선택 사항) 네트워크 볼륨 지정.
   - ✓ 인스턴스 유형 선택 (예: A40).
   - ✓ (선택 사항) 템플릿 제공 (예: RunPod Pytorch).
   - ✓ (GPU만 해당) 컴퓨팅 수 지정.
4 . 구성을 검토하고 Deploy On-Demand를 선택합니다.

Tip : RunPod는 사용자 정의 Dockerfile을 지정할 수 있는 사용자 정의 템플릿을 지원합니다. Dockerfile을 생성함으로써 특정 종속성과 구성이 포함된 사용자 정의 Docker 이미지를 빌드할 수 있습니다. 이를 통해 애플리케이션이 다양한 환경에서 신뢰할 수 있고 이식 가능하도록 보장합니다.

Pod 빌드가 완료된 후에 요금이 발생합니다.

Command line

```
runpodctl create pods \
  --name hello-world \
  --gpuType "NVIDIA A40" \
  --imageName "runpod/pytorch:3.10-2.0.0-117" \
  --containerDiskSize 10 \
  --volumeSize 100 \
  --args "bash -c 'mkdir /testdir1 && /start.sh'"
```

Pod 중지하기

- **웹**

1. 중지 아이콘을 클릭합니다.
2. Stop Pod 버튼을 클릭하여 확인합니다.

- **CLI**

특정 시간 후에 Pod를 중지할 수도 있습니다. 예를 들어, 다음 명령은 2시간 동안 대기한 후 Pod를 중지합니다.

```
sleep 2h; runpodctl stop pod $RUNPOD_POD_ID &
```

이 명령은 2시간 대기한 후 **runpodctl stop pod** 명령을 실행하여 Pod를 중지합니다. 끝의 & 는 명령을 백그라운드에서 실행하여 SSH 세션을 계속 사용할 수 있게 합니다.

※ **경고** : 유휴 Pods를 저장하는 데 요금이 발생합니다. Pod을 저장할 필요가 없다면, 다음에 반드시 종료해야 합니다.

Pod 시작하기

중지된 Pod를 다시 시작할 수 있습니다.

- **웹**

1. Pods 페이지로 이동합니다.
2. 재개할 Pod를 선택합니다.
3. Start를 선택합니다.
4. Pod가 재개됩니다.

- **CLI**

```
runpodctl start pod $RUNPOD_POD_ID
```

Pod 종료하기

※ **위험 : Pod를 종료하면 네트워크 볼륨 외부의 모든 데이터가 영구적으로 삭제됩니다. 다시 접근할 데이터를 저장했는지 확인하세요.**

- **웹**

1. 종료할 Pod의 아래쪽 햄버거 메뉴를 선택합니다.
2. Terminate Pod를 클릭합니다.
3. Yes 버튼을 클릭하여 확인합니다.

- **CLI**

Pod를 제거하기 위해 아래의 커맨드 입력.

```
runpodctl remove pod $RUNPOD_POD_ID
```

Pods 목록 보기

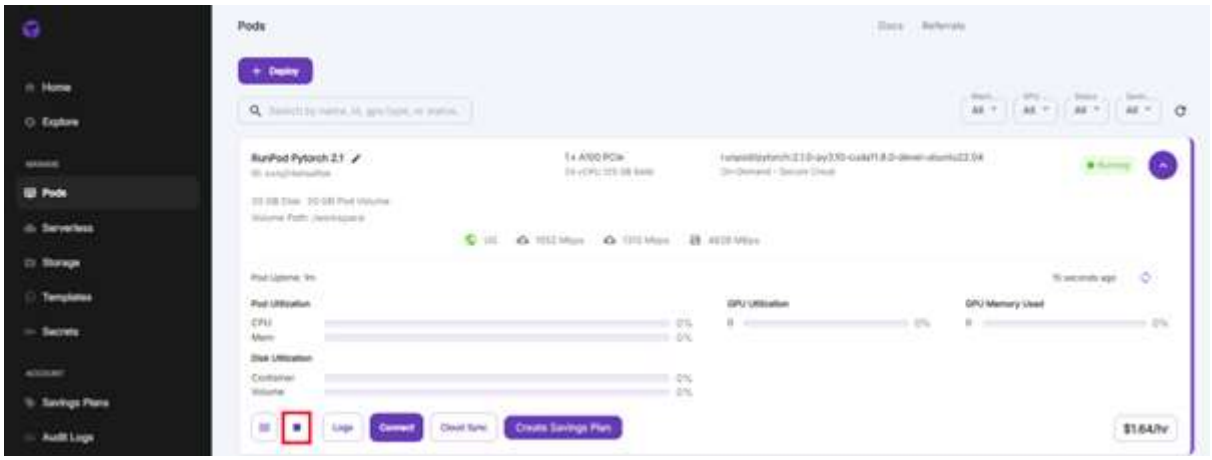
명령줄을 사용하는 경우, 다음 명령을 입력하여 Pods 목록을 확인할 수 있습니다.

```
runpodctl get pod
```

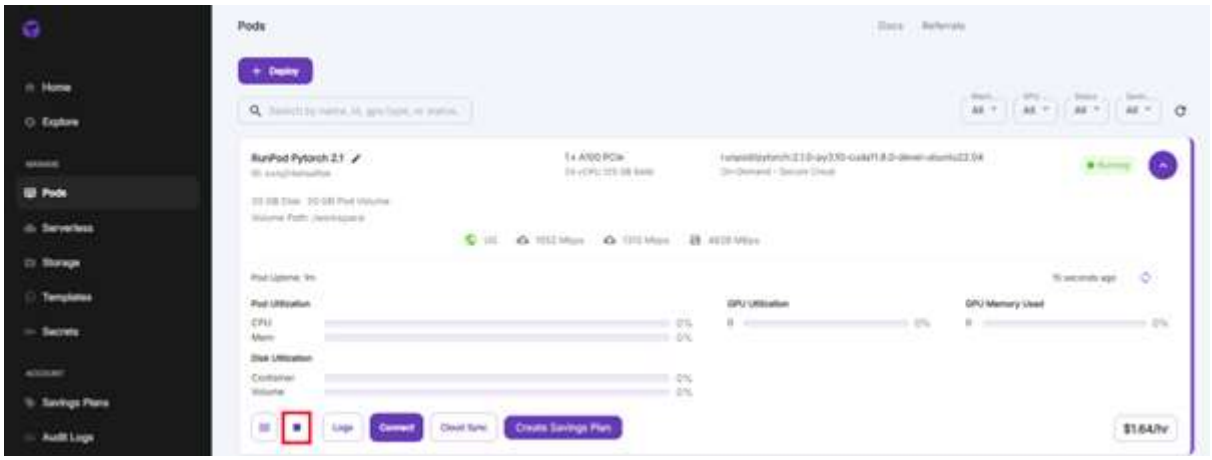
이 지침을 따라 RunPod에서 Pods를 효과적으로 관리하세요!

Pod 제거 간단 요약

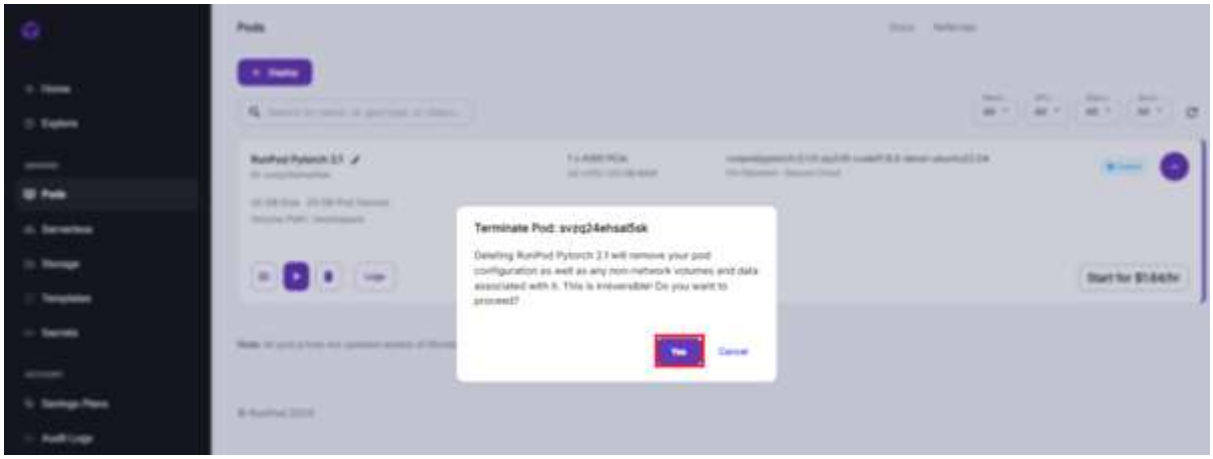
1) 실행 중인 pod을 선택 후 중지 버튼 클릭.



2) 중지 이후 휴지통 버튼 클릭.



3) Yes를 눌러 pod 완전 제거.



4. SDKs

Overview

RunPod SDK는 개발자들이 RunPod API를 사용하여 서버리스 함수 생성 및 인프라 관리를 할 수 있도록 돕는 도구를 제공합니다. 이를 통해 사용자 정의 로직을 통합하고, 배포를 간소화하며, 프로그래밍 방식으로 인프라를 관리할 수 있습니다.

서버리스 엔드포인트와 상호작용하기

배포 후, 서버리스 함수는 엔드포인트로 노출됩니다. 이를 통해 외부 애플리케이션이 HTTP 요청을 통해 서버리스 함수와 상호작용할 수 있습니다.

- **서버리스 엔드포인트는 HTTP 요청처럼 작동합니다.**
- **엔드포인트 ID와 API 키에 대한 참조를 제공하여 요청을 완료할 수 있습니다.**

인프라 관리

RunPod SDK는 Pods, 템플릿, 엔드포인트 등 다양한 인프라 구성 요소의 프로그래밍적 생성, 구성 및 관리를 지원합니다.

- **Pods 관리하기:** Pods는 RunPod에서 응용 프로그램을 실행하기 위한 격리된 환경을 나타내는 기본적인 빌딩 블록입니다.

1. Pod 생성: SDK를 사용하여 원하는 구성으로 새로운 Pod을 인스턴스화합니다.
2. Pod 구성: GPU, 메모리 할당, 네트워크 접근 등의 설정을 필요에 맞게 조정합니다.
3. 응용 프로그램 배포: Pod 내에서 애플리케이션 또는 서비스를 배포합니다.
4. 모니터링 및 확장: SDK를 활용하여 Pod 성능을 모니터링하고 필요에 따라 리소스를 확장합니다.

템플릿과 엔드포인트 관리

템플릿은 Pod의 기본 환경을 정의하며, 엔드포인트는 Pod 내에서 실행 중인 서비스에 외부에서 접근할 수 있도록 합니다.

- **템플릿과 엔드포인트 사용하기**

1. 템플릿 생성: Pod의 기본 구성을 지정하는 템플릿을 정의합니다.

2. 템플릿으로부터 Pod 인스턴스화: 템플릿을 사용하여 일관된 환경을 갖춘 Pod을 생성합니다.
3. 엔드포인트를 통한 서비스 공개: 엔드포인트를 설정하여 Pod 내에서 실행 중인 애플리케이션에 외부 접근을 허용합니다.

Python

Python을 사용하여 RunPod 프로젝트 설정을 시작하는 방법에 대해 설명합니다. 프로젝트의 특정 요구 사항에 따라 RunPod 플랫폼과 상호작용하는 다양한 방법이 있습니다. 이 가이드는 RunPod 플랫폼을 시작하고 실행하는 방법을 제공합니다.

RunPod SDK 설치

RunPod SDK 라이브러리를 설치하려면 Python 가상 환경을 만들어야 합니다. 가상 환경은 각 프로젝트의 종속성을 독립적으로 관리할 수 있게 해주며, 프로젝트 요구 사항 간의 충돌을 방지합니다.

시작하려면 가상 환경을 설정하고, 그 후 RunPod SDK 라이브러리를 설치합니다.

- **venv를 사용하여 Python 가상 환경을 생성합니다.**

```
python3 -m venv env
source env/bin/activate
```

- **SDK를 설치하려면, 터미널에서 아래 명령어를 실행하세요**

```
python -m pip install runpod
```

이제 RunPod SDK가 설치되어 사용 준비가 완료됩니다.

RunPod SDK 버전 확인

설치한 RunPod SDK 버전이 올바르게 설정되었는지 확인하려면, 아래의 방법 중 하나를 사용하여 터미널에서 RunPod Python SDK 버전을 출력합니다.

- **Pip**

터미널에서 다음 명령어를 실행하여 SDK 버전을 확인할 수 있습니다.

```
pip show runpod
```

출력 예시는 다음과 비슷하게 나옵니다:

```
runpod==1.6.1
```

RunPod Python SDK의 최신 버전은 GitHub에서 확인할 수 있습니다.
이제 RunPod SDK가 설치되고 구성되었습니다. API 키를 추가해봅시다.

API 키 추가

API 키를 설정하고 Python 애플리케이션에서 이 변수를 참조하여 요청을 인증합니다. 이를 통해 RunPod 플랫폼에 접근하고 RunPod API를 사용할 수 있습니다.

```
import runpod
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")
```

참고: API 키는 환경 변수로 설정하는 것이 권장됩니다. API 키를 코드에 직접 삽입하는 것은 피해야 합니다.

위 예제에서는 RUNPOD_API_KEY라는 환경 변수에서 API 키를 불러옵니다.

이제 RunPod Python SDK가 설치되고 구성되었으므로, RunPod 플랫폼을 사용하여 다양한 작업을 시작할 수 있습니다.

APIs

- **API Wrapper**

이번에는 RunPod API에서 제공하는 핵심 기능들을 설명합니다. 이를 통해 Endpoints와 템플릿을 관리하고 사용 가능한 GPU 목록을 조회하는 방법을 배울 수 있습니다. 이러한 작업을 통해 RunPod 환경 내에서 컴퓨팅 자원을 동적으로 관리할 수 있습니다.

- **Endpoints 조회**

RunPod 내에서 사용 가능한 모든 엔드포인트 구성을 확인하려면 get_endpoints() 함수를 사용하면 됩니다. 이 함수는 엔드포인트 구성을 목록으로 반환하며, 이를 통해 프로젝트에 사용할 수 있는 엔드포인트들을 파악할 수 있습니다.

```python
import runpod
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

# Fetching all available endpoints
endpoints = runpod.get_endpoints()

# Displaying the list of endpoints
print(endpoints)
```

- **템플릿 생성**

RunPod에서 템플릿은 환경을 효율적으로 설정할 수 있도록 미리 정의된 구성입니다. create_template() 함수를 사용하여 새 템플릿을 생성할 수 있습니다. 템플릿을 생성할 때는 템플릿의 이름과 사용할 Docker 이미지를 지정해야 합니다.

```python
import runpod
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

try:
    # Creating a new template with a specified name and Docker image
    new_template = runpod.create_template(name="test", image_name="runpod/base:0.1.0")

    # Output the created template details
    print(new_template)

except runpod.error.QueryError as err:
    # Handling potential errors during template creation
    print(err)
    print(err.query)
```

- **엔드포인트 생성**

새로운 엔드포인트를 생성하려면 create_endpoint() 함수를 사용합니다. 이 함수는 엔드포인트의 이름과 사용할 템플릿 ID를 지정해야 합니다. 또한 필요에 따라 GPU, 워커 수 등 추가 구성을 할 수 있습니다.

```python
import runpod
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

try:
    # Creating a template to use with the new endpoint
    new_template = runpod.create_template(
        name="test", image_name="runpod/base:0.4.4", is_serverless=True
    )

    # Output the created template details
    print(new_template)

    # Creating a new endpoint using the previously created template
    new_endpoint = runpod.create_endpoint(
        name="test",
        template_id=new_template["id"],
        gpu_ids="AMPERE_16",
        workers_min=0,
        workers_max=1,
    )

    # Output the created endpoint details
    print(new_endpoint)

except runpod.error.QueryError as err:
    # Handling potential errors during endpoint creation
    print(err)
    print(err.query)
```

- **GPU 조회**

사용 가능한 컴퓨팅 자원에 대한 이해를 돕기 위해 get_gpus() 함수를 사용하면 RunPod에서 엔드포인트에 할당 가능한 모든 GPU 목록을 확인할 수 있습니다. 이를 통해 프로젝트에 적합한 GPU를 선택할 수 있습니다.

```python
import runpod
import json
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

# Fetching all available GPUs
gpus = runpod.get_gpus()

# Displaying the GPUs in a formatted manner
print(json.dumps(gpus, indent=2))
```

- **GPU ID로 조회**

특정 GPU 모델에 대한 세부 정보를 확인하려면 get_gpu() 함수를 사용하고, GPU ID를 전달하면 됩니다. 이를 통해 다양한 GPU 모델의 성능과 비용을 확인할 수 있습니다.

```python
import runpod
import json
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

gpus = runpod.get_gpu("NVIDIA A100 80GB PCIe")

print(json.dumps(gpus, indent=2))
```

Endponit

이번에는 RunPod Python SDK를 사용하여 다양한 엔드포인트와 상호작용하는 방법을 설명합니다. 이를 통해 동기 및 비동기 작업을 처리하고, 데이터를 스트리밍하며, 엔드포인트의 상태를 확인할 수 있습니다.

- **필수 조건**

RunPod Python SDK를 사용하기 전에 아래 사항을 준비해야 합니다.

1. RunPod Python SDK 설치
2. API 키 설정

- **엔드포인트 ID 설정**

엔드포인트 클래스에 엔드포인트 ID를 전달합니다.

```python
import runpod
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

endpoint = runpod.Endpoint("YOUR_ENDPOINT_ID")
```

이렇게 하면 모든 호출이 유효한 API 키로 Endpoint ID를 통과할 수 있습니다. 대부분의 상황에서 엔드포인트 클래스에 변수 이름 엔드포인트를 설정할 수 있습니다. 이렇게 하면 엔드포인트 클래스의 다음 방법 또는 인스턴스 변수를 사용할 수 있습니다.

- **엔드포인트 실행**

엔드포인트는 비동기(run) 또는 동기(run_sync) 방식으로 실행할 수 있습니다.
비동기 및 동기 실행 방식의 선택은 작업의 요구 사항과 애플리케이션 설계에 따라 달라집니다.

비동기 방식을 선택하면 작업을 효율적으로 처리할 수 있습니다. 특히 즉각적인 피드백이 중요하지 않은 경우에 유리합니다. 비동기 실행 방식은 애플리케이션이 장시간 작업을 처리하는 동

안에도 응답 상태를 유지할 수 있게 해줍니다. 이는 다음과 같은 상황에 적합합니다.

- ✓ **비차단 호출**: 긴 처리 시간이 필요한 작업을 기다리는 동안 애플리케이션이 계속 활성 상태를 유지할 수 있습니다.
- ✓ **긴 작업 처리**: 30초 이상의 작업에 대해서 타임아웃을 방지하고, 애플리케이션 흐름을 계속해서 유지할 수 있습니다.
- ✓ **작업 추적:** 작업 ID를 통해 작업 상태를 모니터링할 수 있어, 복잡하거나 결과가 지연되는 작업에 유용합니다.

동기 방식을 선택하면 작업이 완료될 때까지 기다릴 수 있습니다. 즉시 결과가 필요한 경우에 유리합니다. 동기 실행 방식은 다음과 같은 상황에서 적합합니다.

- ✓ **즉각적인 결과**: 애플리케이션의 논리를 계속 진행하기 위해 빠른 결과가 필요한 작업에 적합합니다.
- ✓ **짧은 작업**: 30초 이하의 짧은 작업을 처리하는 데 이상적입니다.
- ✓ **단순함과 제어**: 실행 과정이 간단하며, 타임아웃 설정을 통해 더 나은 작업 제어가 가능합니다.

- **동기식 실행**

엔드포인트를 동기식으로 실행하고 결과를 기다리려면 run_sync 메서드를 사용하세요. 이 메서드는 엔드포인트 실행이 완료되거나 타임아웃이 발생할 때까지 실행을 차단합니다.

```python
import runpod
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

endpoint = runpod.Endpoint("YOUR_ENDPOINT_ID")

try:
    run_request = endpoint.run_sync(
        {
            "input": {
                "prompt": "Hello, world!",
            }
        },
        timeout=60,  # Timeout in seconds.
    )

    print(run_request)
except TimeoutError:
    print("Job timed out.")
```

- **비동기식 실행**

비동기식 실행은 차단되지 않는 작업을 가능하게 하여, 작업이 완료될 때까지 기다리는 동안 다른 작업을 수행할 수 있도록 합니다. RunPod는 표준 비동기 실행뿐만 아니라 Python의 asyncio 프레임워크를 이용한 고급 비동기 프로그래밍도 지원합니다.

애플리케이션의 요구 사항에 따라 가장 적합한 방법을 선택할 수 있습니다.

비차단 작업을 위해 run 메서드를 사용하세요. 이 메서드는 엔드포인트 실행을 시작한 후, 상태를 확인하거나 나중에 완료를 기다릴 수 있도록 합니다.

이 방식은 비동기 이벤트 루프 없이 표준 Python 환경에서 실행됩니다.

```python
import runpod
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

input_payload = {"input": {"prompt": "Hello, World!"}}

try:
    endpoint = runpod.Endpoint("YOUR_ENDPOINT_ID")
    run_request = endpoint.run(input_payload)

    # Initial check without blocking, useful for quick tasks
    status = run_request.status()
    print(f"Initial job status: {status}")

    if status != "COMPLETED":
        # Polling with timeout for long-running tasks
        output = run_request.output(timeout=60)
    else:
        output = run_request.output()
    print(f"Job output: {output}")
except Exception as e:
    print(f"An error occurred: {e}")
```

- **asyncio를 이용한 비동기식 실행**

Python의 asyncio 라이브러리를 사용하여 엔드포인트 호출을 효율적으로 처리할 수 있습니다. 이 방식은 Python의 asyncio 프레임워크를 활용한 비동기 프로그래밍으로, 함수는 async로 정의하고 await를 사용하여 호출해야 합니다. 이 접근 방식은 본래 차단되지 않으며, 동시성을 효율적으로 처리할 수 있도록 설계되었습니다.

```python
import asyncio
import aiohttp
import os
import runpod
from runpod import AsyncioEndpoint, AsyncioJob

# asyncio.set_event_loop_policy(asyncio.WindowsSelectorEventLoopPolicy())  # For Windows users.


runpod.api_key = os.getenv("RUNPOD_API_KEY")


async def main():
    async with aiohttp.ClientSession() as session:
        input_payload = {"prompt": "Hello, World!"}
        endpoint = AsyncioEndpoint("YOUR_ENDPOINT_ID", session)
        job: AsyncioJob = await endpoint.run(input_payload)

        # Polling job status
        while True:
            status = await job.status()
            print(f"Current job status: {status}")
            if status == "COMPLETED":
                output = await job.output()
                print("Job output:", output)
                break  # Exit the loop once the job is completed.
            elif status in ["FAILED"]:
                print("Job failed or encountered an error.")

                break
            else:
                print("Job in queue or processing. Waiting 3 seconds...")
                await asyncio.sleep(3)  # Wait for 3 seconds before polling again


if __name__ == "__main__":
    asyncio.run(main())
```

- **헬스 체크**

엔드포인트의 상태를 모니터링하여 완료된 작업, 실패한 작업, 진행 중인 작업, 대기 중인 작업 및 재시도된 작업뿐만 아니라 워커(worker)의 상태도 확인할 수 있습니다.

```python
import runpod
import json
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

endpoint = runpod.Endpoint("gwp4kx5yd3nur1")

endpoint_health = endpoint.health()

print(json.dumps(endpoint_health, indent=2))
```

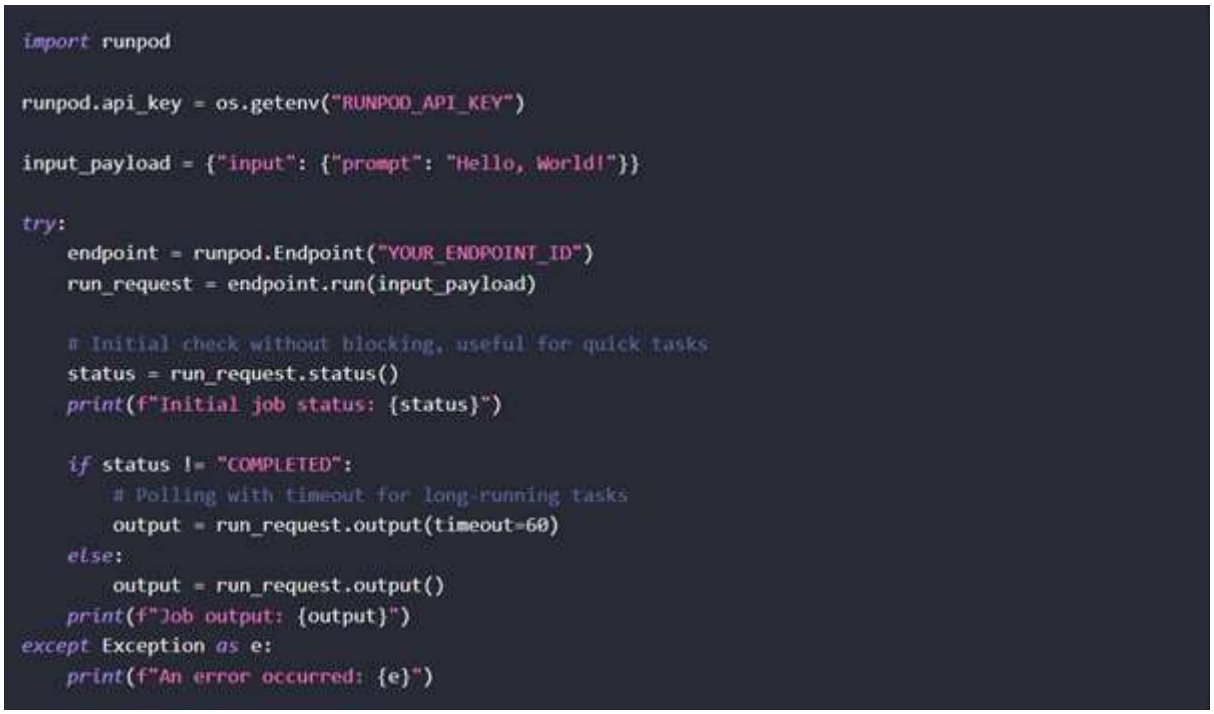
- **스트리밍**

스트리밍을 활성화하려면 핸들러의 start 메서드에서 "return_aggregate_stream": True 옵션을 지원해야 합니다. 스트리밍이 활성화되면, stream 메서드를 사용하여 데이터가 준비되는 대로 받을 수 있습니다.

```python
import runpod

runpod.api_key = os.getenv("RUNPOD_API_KEY")

endpoint = runpod.Endpoint("YOUR_ENDPOINT_ID")

run_request = endpoint.run(
    {
        "input": {
            "prompt": "Hello, world!",
        }
    }
)

for output in run_request.stream():
    print(output)
```

- **상태**

작업 요청의 상태를 반환합니다. run 요청에서 status() 함수를 설정하여 작업의 상태를 반환하도록 할 수 있습니다.



```python
import runpod

runpod.api_key = os.getenv("RUNPOD_API_KEY")

input_payload = {"input": {"prompt": "Hello, World!"}}

try:
    endpoint = runpod.Endpoint("YOUR_ENDPOINT_ID")
    run_request = endpoint.run(input_payload)

    # Initial check without blocking, useful for quick tasks
    status = run_request.status()
    print(f"Initial job status: {status}")

    if status != "COMPLETED":
        # Polling with timeout for long-running tasks
        output = run_request.output(timeout=60)
    else:
        output = run_request.output()
    print(f"Job output: {output}")
except Exception as e:
    print(f"An error occurred: {e}")
```

- **취소**

run 요청에서 cancel() 함수를 사용하여 작업 요청을 취소할 수 있습니다. 작업이 IN_QUEUE 또는 IN_PROGRESS 상태에 갇혀 있거나 더 이상 결과가 필요 없을 때 작업을 취소하고자 할 수 있습니다.

다음 패턴은 사용자의 상호작용에 따라 작업을 취소하는 예시입니다. 예를 들어, 터미널에서 Ctrl+C를 눌렀을 때 작업을 취소할 수 있습니다.

이 방법은 KeyboardInterrupt 예외를 처리하여 실행 중인 작업에 SIGINT 신호를 전송합니다.

```python
import time
import runpod

runpod.api_key = os.getenv("RUNPOD_API_KEY")

input_payload = {
    "messages": [{"role": "user", "content": f"Hello, World"}],
    "max_tokens": 2048,
    "use_openai_format": True,
}

try:
    endpoint = runpod.Endpoint("YOUR_ENDPOINT_ID")
    run_request = rp_endpoint.run(input_payload)

    while True:
        status = run_request.status()
        print(f"Current job status: {status}")

        if status == "COMPLETED":
            output = run_request.output()
            print("Job output:", output)

            generated_text = (
                output.get("choices", [{}])[0].get("message", {}).get("content")
            )
            print(generated_text)
            break
        elif status in ["FAILED", "ERROR"]:
            print("Job failed to complete successfully.")
            break
        else:
            time.sleep(10)
except KeyboardInterrupt:  # Catch KeyboardInterrupt
    print("KeyboardInterrupt detected. Canceling the job...")
    if run_request:  # Check if a job is active
        run_request.cancel()
    print("Job canceled.")

except Exception as e:
    print(f"An error occurred: {e}")
```

- **타임아웃**

cancel() 함수와 timeout 인자를 사용하여 지정된 시간 후에 작업을 취소할 수 있습니다. 앞서 설명한 cancel() 예시에서는 외부 조건에 의해 작업이 취소되었습니다. 이 예시에서는 완료하는 데 너무 오랜 시간이 걸린 실행 중인 작업을 취소하는 방법을 설명합니다.

```python
from time import sleep
import runpod
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

input_payload = {"input": {"prompt": "Hello, World!"}}

endpoint = runpod.Endpoint("YOUR_ENDPOINT_ID")


# Submit the job request
run_request = endpoint.run(input_payload)

# Retrieve and print the initial job status
initial_status = run_request.status()
print(f"Initial job status: {initial_status}")

# Attempt to cancel the job after a specified timeout period (in seconds)
# Note: This demonstrates an immediate cancellation for demonstration purposes.
# Typically, you'd set the timeout based on expected job completion time.
run_request.cancel(timeout=3)

# Wait for the timeout period to ensure the cancellation takes effect
sleep(3)
print("Sleeping for 3 seconds to allow for job cancellation...")

# Check and print the job status after the sleep period
final_status = run_request.status()
print(f"Final job status: {final_status}")
```

- **큐 정리**

purge_queue() 함수를 사용하여 큐에서 모든 작업을 정리할 수 있습니다. timeout 파라미터를 제공하여 서버가 응답할 때까지 기다릴 시간을 지정할 수 있습니다.

purge_queue()는 진행 중인 작업에는 영향을 미치지 않습니다.

```python
import runpod
import os

runpod.api_key = os.getenv("RUNPOD_API_KEY")

endpoint = runpod.Endpoint("YOUR_ENDPOINT_ID")

endpoint.purge_queue(timeout=3)
```

5. Hosting

Overview

- **RunPod GPU 호스팅 기회**

RunPod는 고유한 서버와 신뢰할 수 있는 커뮤니티 구성원들과의 협업을 통해 다양한 GPU 옵션을 제공합니다. 만약 자신의 하드웨어를 RunPod 생태계에 통합하고자 한다면, 아래 단계를 따라 주세요.

- **호스트로 가입하는 방법**

1. **자격 확인**: 최소 요구 사항을 충족하는지 확인하세요.
2. **연락하기**: 현재는 호스트를 수동으로 심사하여 온보딩하고 있습니다. 고품질의 기계가 있고 총 20개의 GPU 이상을 보유하고 있다면, 이 양식을 작성해 주세요.

- **추가 호스팅 정보**

1. 서비스 수수료: RunPod는 24%의 서비스 수수료를 부과합니다. 이는 다음을 포함합니다.
   - 약 4%는 Stripe 결제 수수료
   - 2%는 추천 프로그램을 위한 수수료
   - 1%는 템플릿 프로그램을 위한 수수료입니다. 자세한 사항은 "친구 추천"을 참조하세요.

2. 가격 책정: GPU 온디맨드 가격은 일정하지만, 호스트는 스팟 렌탈의 최소 입찰가를 설정할 수 있습니다. 우리는 가능한 한 가격을 안정적으로 유지하려고 노력하지만, 시장의 변동성에 맞춰 조정할 필요가 있습니다.

3. 안전성 및 신뢰성: 우리는 모든 호스트에게 KYC(고객 확인) 인증을 의무화하여 사용자 보호와 사기 방지에 힘쓰고 있습니다. 대형 제공자의 경우, 제공자 계약서와 서비스 수준 계약(SLA)을 작성해야 합니다.

4. 호스팅 경험: 신뢰할 수 있는 제공자로서, 리소스를 관리할 수 있는 완전히 커스터마이즈된 대시보드에 액세스할 수 있으며, 이를 통해 하드웨어를 배포하고 확장 계획을 수립할 수 있습니다.

5. 렌탈 요금: 우리는 사용률 데이터를 공개하지 않지만, 특정 GPU 모델에 대한 인기와 통계는 직접 논의 시 제공할 수 있습니다. 또한 다양한 변수가 점유율에 영향을 미칠 수 있으므로, 하드웨어 품질 수준에 따라 수익에 어떤 영향을 미칠지에 대한 심층 데이터를 제공할 준비가 되어 있습니다.

Burn Testing

RunPod에 기계를 등록하기 전에 철저한 테스트가 필요합니다. 아래는 며칠 동안 수행할 수 있는 간단한 burn 테스트 가이드입니다.

1. RunPod 에이전트 중지하기

먼저, RunPod 에이전트를 중지해야 합니다. 다음 명령어를 실행하세요.

```
sudo systemctl stop runpod
```

2. GPU-Burn 실행하기

그 후, GPU burn 테스트를 시작하려면 다음 명령어를 입력하세요.

```
docker run --gpus all --rm jorghi21/gpu-burn-test 172800
```

위 명령어에서 172800은 48시간(초 단위) 동안 테스트를 실행하겠다는 의미입니다. 테스트 시간은 필요에 따라 조정할 수 있습니다.

3. 시스템 리소스 확인하기

메모리, CPU, 디스크가 충분한지 확인해야 합니다. 이를 위해 ngstress 라이브러리를 사용할 수 있습니다. ngstress는 CPU와 메모리 부하를 테스트하는 데 유용한 도구입니다.

4. RunPod 에이전트 다시 시작하기

모든 검증이 완료되면, RunPod 에이전트를 다시 시작합니다:

```
sudo systemctl start runpod
```

5. 기계 대시보드에서 테스트하기

기계가 제대로 작동하는지 확인하려면, 자신의 기계 대시보드에서 자체 렌탈을 해보세요. 이를 통해 가장 인기 있는 템플릿에서 잘 작동하는지 테스트할 수 있습니다.

Maintenance and reliability

- **유지보수 안내**

호스트는 유지보수를 최소 1주일 전에 예약해야 하며, 서버가 사용되지 않은 경우에만 즉시 유지보수를 진행할 수 있습니다. 유지보수가 예정되어 있는 경우, 활성화된 Pod에 대한 사용자들에게 이메일 알림이 발송됩니다. 만약 아래와 같은 상황이라면, Discord나 Slack을 통해 RunPod에 연락해주세요.

○ 여러 대의 기계에 대해 유지보수를 예약하는 경우
○ 사용자 데이터에 영향을 미칠 수 있는 작업을 수행하는 경우

항상 조심스럽게 행동하고, 의사소통을 과도하게 하는 것을 목표로 하세요.

- **유지보수 시 유의 사항**

1. 예정된 유지보수 중에는 가동 시간/신뢰성에 영향이 없습니다.
예약된 유지보수 중에는 고객의 작업에 영향을 주지 않도록 합니다.

2. 기타 이벤트로 인한 신뢰성 점수 감소
예약되지 않은 유지보수 또는 이벤트가 발생하면 신뢰성 점수가 낮아집니다. 이는 목록에 없는 기계에도 해당됩니다.

3. 유지보수 예약 시 기계 자동 목록 제외
유지보수가 예약된 모든 기계는 고객의 작업에 방해가 되지 않도록 예약된 유지보수 시작 4일 전에 자동으로 목록에서 제외됩니다.

4. 과도한 유지보수는 추가 처벌을 초래합니다.
과도한 유지보수나 불필요한 작업은 신뢰성에 부정적인 영향을 미치고, 추가적인 페널티가 부과될 수 있습니다.

5. 활성 사용자가 있는 기계의 유지보수
유지보수 창 내에서만 활성 사용자가 있는 기계를 내려놓을 수 있습니다. 다만, 해당 기계에서의 작업이 사용자 데이터에 영향을 미칠 수 있으므로 주의가 필요합니다.

6. 즉시 유지보수
즉시 유지보수는 긴급한 수리/업데이트가 필요한 경우에만 사용해야 합니다. 사용되지 않는 서버는 여전히 사용자 데이터를 보관할 수 있으며, 데이터 손실을 초래할 수 있는 작업은 이 모드에서 진행하지 말아야 합니다.

신뢰성 계산

RunPod는 99.99%의 가동 시간을 제공하는 데이터 센터와 협력하는 것을 목표로 하고 있습니다. 신뢰성은 다음과 같이 계산됩니다.

신뢰성=총 시간+소형 버퍼인터벌의 총 시간₩text{신뢰성} = ₩frac{₩text{총 시간} + ₩text{소형 버퍼}}{₩text{인터벌의 총 시간}}신뢰성=인터벌의 총 시간총 시간+소형 버퍼

예를 들어, 한 달의 첫날에 30분의 네트워크 다운타임이 발생했다고 가정하면, 신뢰성은 다음과 같이 계산됩니다.

신뢰성=43200−30+1043200=99.95%₩text{신뢰성} = ₩frac{43200 - 30 + 10}{43200} = 99.95₩%신뢰성=4320043200−30+10=99.95%

- 43200분은 한 달의 총 분 수(30일 기준)입니다.
- **소형 버퍼(10분)**는 가끔 발생하는 작은 다운타임을 고려한 것입니다.
- 한 달 내내 다운타임이 없다면, 100%의 신뢰성을 유지하려면 한 달이 걸립니다.

신뢰성 점수가 98% 이하인 기계는 자동으로 GPU 풀에서 제외되며, 해당 기계는 이미 데이터가 있는 고객만 접근할 수 있습니다.

RunPod 호스팅 요구사항

아래의 요구 사항은 최소 조건이며, 향후 변경될 수 있습니다. 이 조건들은 GPU 호스팅을 위한 필수 기준을 다루고 있습니다.

< 소프트웨어 사양 >

- **운영 체제**

  Ubuntu Server 22.04 LTS

  - 기본적인 리눅스 사용 능력
  - 원격으로 SSH 연결할 수 있는 능력

- **Ubuntu Server 22.04 LTS 설치**

  22.04 파일을 사용하되, 설치 시 HWE(Hardware Enablement) 옵션을 선택하여 Kernel 6.5.0-15가 설치되도록 합니다. (더 최신 버전이 있다면 해당 버전으로 교체)

- **BIOS 설정**

  비가상화 시스템에서 IOMMU를 비활성화 해야 합니다.

  호환성 문제를 겪고 있다면 서버의 BIOS를 최신 안정 버전으로 업데이트하는 것이 좋습니다.

- **드라이버**

  Nvidia 드라이버: 550.54.15 (더 최신 버전으로 교체 가능)

  CUDA: 12.4 (더 최신 버전으로 교체 가능)

  Nvidia Persistence: 48 GB 이상의 GPU에 대해 활성화해야 합니다.

- **HGX SXM 시스템**
  - Nvidia Fabric Manager 설치 및 활성화 필요
  - Fabric Manager 버전, Nvidia 드라이버 버전, Kernel 드라이버 헤더는 모두 일치해야 합니다.
  - P2P 대역폭 테스트를 통과해야 하며, CUDA Toolkit, Nvidia NSCQ, Nvidia DCGM도 설치되어야 합니다.
  - NVLink 스위치의 토폴로지를 확인하기 위해 nvidia-smi와 dcgmi를 활용합니다.
  - SXM 성능 확인을 위해 dcgmi 진단 도구를 사용합니다.

< 최소 보안 클라우드 사양 >

- **GPU 모델**
  - 최신 Nvidia GPU 모델이 필요하며, 최소한 30xx 시리즈 또는 RTX A4000 이상의 모델이 요구됩니다.
  - 높은 수요: SXM 80 GB, PCIe 80 GB, Ada 6000 48 GB, Ada 5000 32 GB, 4090 24 GB, L4 24 GB, A5000 24 GB, Ada 4000

- **GPU 수량**
  - 옵션 1: 총 100개 이상의 GPU, 각 GPU는 최소 12 GB VRAM 필요
  - 옵션 2: 총 32개 이상의 GPU, 각 GPU는 최소 80 GB VRAM 필요
  - 최소 1 서버당 2개 이상의 GPU 필요, 8x GPU 구성이 권장됨

- **CPU**
  - GPU당 최소 4개의 물리적 CPU 코어와 시스템 운영을 위한 2개 이상의 코어 필요
  - CPU의 클럭 속도가 더 빠른 것이 중요 (예: 24코어 5.0 GHz CPU가 128코어 3.0 GHz CPU보다 우선시됨)
  - Genoa CPU가 이런 요구 사항에 적합할 수 있음

- **Bus Bandwidth**
  - 8 GB, 10 GB, 12 GB, 16 GB GPU에는 최소 PCIe 3.0 x16 필요
  - 20 GB, 24 GB, 32 GB, 40 GB, 48 GB, 80 GB GPU에는 최소 PCIe 4.0 x16 필요
  - 80 GB GPU에는 PCIe 5.0 x16이 권장됨

- **메모리**
  - 시스템 RAM은 전체 VRAM + 12 GB 이상이어야 합니다.
  - 8x 80 GB GPU 구성에는 최소 1024 GB RAM 권장
  - 8x 24 GB GPU 구성에는 최소 512 GB RAM 권장
  - 8x 16 GB GPU 구성에는 최소 256 GB RAM 권장
  - DDR4는 최소 요구 사항, DDR5가 권장됨
  - ECC(Error-Correcting Code) 호환 메모리 필요

- **저장소**
  - GPU당 최소 1 TB 이상의 NVME 저장소 필요
    24 GB 및 48 GB GPU에는 2 TB 이상의 NVME 추천
    80 GB GPU에는 4 TB 이상의 NVME 추천
  - 운영 체제는 RAID 1으로 구성된 2개의 작은 NVME 디스크(500 GB 또는 1 TB 추천)
  - 데이터 디스크는 LVM 또는 RAID로 설정되어야 하며, 여러 데이터 드라이브가 있을 경우 RAID 10 구성이 이상적
  - 읽기/쓰기 속도는 3,000 MBps 이상 필요

- PCIe 5.0 x4 NVME SSD는 80 GB 및 최신 48 GB GPU에 유리함
- 네트워크 스토리지 클러스터 배포 가능해야 함

- **네트워킹**
  - 10 Gbps 양방향 인터넷 속도를 지원하는 백본
  - 서버당 최소 1 Gbps 대칭 인터넷 접속 필요
  - 정적 공인 IP 필요
    공인 IP는 최대 20개의 서버 그룹에 대해 공유 가능
  - 포트 포워딩 가능해야 하며, 30개 이상의 포트가 서버당 필요
  - 서버 간 최소 25 Gbps의 인터커넥트 속도 필요
  - 80 GB GPU에는 200 Gbps의 서버 간 인터커넥트 속도가 권장됨

- **컴플라이언스**
  - Tier III+ 데이터 센터 표준 준수
  - 안정적인 UPS(무정전 전원 공급 장치) 및 백업 발전기
  - 스위치 및 PDU 이중화
  - ISP 이중화
  - 24/7 현장 보안 및 기술 직원
  - 유지보수 및 다운타임은 최소 1주일 전에 예약해야 합니다

< 가장 중요한 요구 사항 >

- **GPU 공급 확장 가능성:** 시간이 지남에 따라 GPU 공급을 확장할 수 있어야 합니다.
- 순수한 거래 관계를 넘어서 AI 클라우드 인프라의 미래를 함께 구축하려는 파트너십에 대한 관심이 필요합니다.

6. References

Trouble Shooting

- **유출된 API 키**

유출된 API 키는 사용자가 실수로 공개된 리포지토리에 평문 API 키를 포함할 때 발생할 수 있습니다. 이 문서는 손상된 키를 복구하는 데 도움이 되는 지침을 제공합니다.

✓ **비활성화**

API 키를 비활성화하려면

1. 콘솔에서 설정을 선택합니다.
2. API 키 섹션에서 토글을 선택하고 예를 선택합니다.

✓ **취소**

API 키를 삭제하려면

1. 콘솔에서 설정을 선택합니다.
2. API 키 섹션에서 휴지통 아이콘을 선택하고 키 취소를 선택합니다.

- **저장 공간 부족**

저장 공간 부족은 사용자가 많은 파일을 생성하거나 파일을 전송하거나 다른 저장 공간을 많이 사용하는 작업을 수행할 때 발생할 수 있습니다. 이 문서는 저장 공간 부족 문제를 해결하는 방법을 안내합니다.

✓ **디스크 사용량 확인**

저장 공간 부족 문제를 겪을 때 첫 번째 단계는 컨테이너의 디스크 사용량을 확인하는 것입니다. df -h 명령어를 사용하여 디스크 사용량 요약을 확인할 수 있습니다.

```
df -h
```

출력 예시:

```
root@9b8e325167b2:/# df -h
Filesystem                Size  Used Avail Use% Mounted on
overlay                    20G   16M   20G   1% /
tmpfs                      64M     0   64M   0% /dev
tmpfs                     252G     0  252G   0% /sys/fs/cgroup
shm                        24G     0   24G   0% /dev/shm
/dev/sda2                 457G   12G  423G   3% /usr/bin/nvidia-smi
tmpfs                     252G   12K  252G   1% /proc/driver/nvidia
tmpfs                     252G  4.0K  252G   1% /etc/nvidia/nvidia-application-profiles-rc.d
tmpfs                      51G  4.4M   51G   1% /run/nvidia-persistenced/socket
tmpfs                     252G     0  252G   0% /proc/asound
tmpfs                     252G     0  252G   0% /proc/acpi
tmpfs                     252G     0  252G   0% /proc/scsi
tmpfs                     252G     0  252G   0% /sys/firmware
tmpfs                     252G     0  252G   0% /sys/devices/virtual/powercap
```

- 확인해야 할 주요 영역
  - ✓ **컨테이너 디스크 사용량**: 컨테이너의 주요 저장 공간은 overlay 파일 시스템에 마운트됩니다. 이는 컨테이너의 루트 디렉토리입니다.

```
Filesystem                Size  Used Avail Use% Mounted on
overlay                    20G   16M   20G   1% /
```

현재 디렉토리의 공간 사용량을 확인하려면 du -sh . 명령어를 사용할 수 있습니다. 컨테이너의 기본 볼륨이나 네트워크 볼륨은 /workspace에 마운트되므로 아래와 같이 확인할 수 있습니다.

```
root@9b8e325167b2:/# cd workspace/
root@9b8e325167b2:/workspace# du -sh .
194M    .
```

  - ✓ **대용량 파일 확인:** /workspace에서 가장 큰 10개의 파일을 찾으려면 다음 명령어를 사용할 수 있습니다.

```
root@9b8e325167b2:/# find /workspace -type f -exec du -h {} + | sort -rh | head -n 10
96M     /workspace/f.txt
96M     /workspace/e.txt
1.0K    /workspace/c.txt
512     /workspace/b.txt
512     /workspace/a.txt
```

✓ **파일 및 디렉토리 삭제**

더 이상 필요 없는 대용량 파일이나 디렉토리를 식별한 후, 이를 삭제하여 공간을 확보할 수 있습니다.

※**경고**: 이 작업은 파일이나 폴더를 영구적으로 삭제합니다. 신중하게 사용하세요.

```
# To delete a specific file, use the rm command:
rm /path/to/file

# To remove an entire directory and its contents, use the rm -r command:
rm -r /path/to/directory
```

자주 묻는 질문 (FAQ)

- **보안 클라우드 vs 커뮤니티 클라우드**

RunPod는 보안 클라우드 (Secure Cloud)와 커뮤니티 클라우드 (Community Cloud)의 두 가지 클라우드 컴퓨팅 서비스를 제공합니다.

보안 클라우드는 신뢰할 수 있는 파트너가 운영하는 T3/T4 데이터 센터에서 실행됩니다. 긴밀한 파트너십을 통해 고도의 신뢰성, 중복성, 보안 및 빠른 응답 시간을 제공하며, 다운타임을 최소화합니다. 민감한 데이터나 기업용 워크로드에는 보안 클라우드를 강력히 추천합니다.

커뮤니티 클라우드는 전 세계의 다양한 사용자가 참여하는 분산 플랫폼을 제공합니다. 이를 통해 개인 컴퓨팅 제공자와 소비자가 서로 연결되어 P2P GPU 컴퓨팅을 이용할 수 있습니다. 커뮤니티 클라우드 호스트는 초대제로 운영되며, 우리가 엄격히 심사를 거친 후 기준을 준수해야 합니다. 비록 인프라의 중복성은 다소 낮을 수 있지만, 품질과 경제성을 결합한 좋은 서버를 제공합니다.

두 서비스 모두 AWS나 GCP와 같은 대형 클라우드 제공업체들보다 훨씬 경쟁력 있는 가격을 제공합니다.

- **온디맨드 vs 스팟 팟**

온디맨드 팟은 중단 없이 계속 실행될 수 있으며, 팟에 할당된 리소스가 전용으로 제공됩니다. 그러나 스팟 팟보다 비용이 더 높습니다.

스팟 팟은 여유 컴퓨팅 용량을 사용하여 리소스를 입찰로 얻을 수 있습니다. 리소스는 당신의 팟에 전용으로 할당되지만, 다른 사용자가 더 높은 가격을 입찰하거나 온디맨드 팟을 시작할 경우 해당 팟은 중단될 수 있습니다. 이때 팟에는 5초 전에 SIGTERM 신호가 전달되고, 그 후 5초 이내에 SIGKILL 신호가 전달됩니다. 이 5초 동안 데이터를 디스크에 저장하거나 클라우드로 주기적으로 전송할 수 있습니다.

- **RunPod가 어떻게 작동하나요?**

RunPod는 Docker와 같은 기술을 활용하여 게스트 워크로드를 호스트 머신에서 컨테이너화하고 격리합니다. 우리는 수천 대의 서버가 연결되어 원활한 사용자 경험을 제공하는 분산 플랫폼을 구축했습니다.

- **어디에서 도움을 받을 수 있나요?**

우리는 기꺼이 도와드립니다! Discord 커뮤니티에 참여하거나, 지원 채팅을 통해 메시지를 보내거나, help@runpod.io로 이메일을 보내 주세요.

- **RunPod의 환불 및 크레딧 정책은 무엇인가요?**

RunPod가 자신에게 적합한지 확실하지 않다면, Discord에서 질문을 하거나 help@runpod.io로 이메일을 보내 주세요. 계정에 최소 $10을 충전하여 시험해 볼 수 있습니다. 현재는 환불이나 시험 크레딧을 제공하지 않으며, 이는 이러한 요청 처리에 드는 관리 비용 때문입니다. 이를 고려하여 계획해 주세요.

- **팟(Pod)이란 무엇인가요?**
  - ✓ **온디맨드 인스턴스**: 중단 없이 실행되는 워크로드에 사용됩니다. 온디맨드 가격을 지불하면 다른 사용자가 이를 중단할 수 없으며, 팟이 계속 실행될 수 있습니다.
  - ✓ **스팟 인스턴스**: 일반적으로 온디맨드 인스턴스보다 훨씬 저렴한 가격으로 임대할 수 있는 중단 가능한 인스턴스입니다. 상태 비저장(workload) API와 같은 워크로드에 적합하며, 주기적으로 데이터를 볼륨에 저장할 수 있습니다. 스팟 인스턴스가 중단되더라도 볼륨은 그대로 유지됩니다.

- **청구 (Billing)**
  - ✓ **팟 청구**: 모든 팟은 GPU 유형에 따라 시간당 요금이 부과됩니다. 팟이 실행 중인 동안 매분마다 크레딧이 차감됩니다. 크레딧이 부족하면 팟이 자동으로 중지되며, 이메일로 알림을 받게 됩니다. 크레딧이 부족한 경우 팟은 종료됩니다.
  - ✓ **저장소 청구**: 현재 실행 중인 팟의 저장소는 월 $0.10/GB, 중지된 팟의 볼륨 저장소는 월 $0.20/GB로 부과됩니다. 저장소는 분 단위로 청구됩니다.
  - ✓ **네트워크 볼륨 청구**: 1TB 이하의 저장소는 월 $0.07/GB의 요금이 부과됩니다. 1TB 이상인 경우, 월 $0.05/GB의 경쟁력 있는 가격으로 제공됩니다.

- **데이터 보호 (Security)**
  - ✓ **내 데이터는 다른 클라이언트로부터 보호되나요?**

예. RunPod는 다중 임차 환경에서 실행되며, 다른 클라이언트는 당신의 팟에 접근할 수 없습니다. 민감한 워크로드는 보안 클라우드를 사용하는 것이 좋습니다.

✓ **내 데이터는 팟이 실행 중인 호스트로부터 보호되나요?**

RunPod의 서비스 약관은 호스트가 당신의 팟 데이터를 확인하거나 사용 패턴을 검사하는 것을 금지합니다. 최고의 보안을 원한다면 보안 클라우드를 사용하세요.

- **사용성 (Usability)**

✓ **RunPod에서 어떤 작업을 할 수 있나요?**

Docker 컨테이너를 실행할 수 있으며, 기본적으로 제공되는 템플릿을 사용하거나 자신만의 템플릿을 만들 수 있습니다.

✓ **내 Docker 데몬을 RunPod에서 실행할 수 있나요?**

현재는 자신만의 Docker 인스턴스를 실행할 수 없습니다. RunPod가 Docker를 대신 실행하므로, Docker 컨테이너를 생성하거나 Docker Compose를 사용하는 등의 작업은 불가능합니다. 대부분의 사용 사례는 실행하고자 하는 Docker 이미지를 담은 커스텀 템플릿을 만들어 해결할 수 있습니다.

✓ **내 팟이 초기화 중에 멈췄어요. 이유가 무엇인가요?**

보통 여러 가지 이유로 초기화가 멈출 수 있습니다. 문제를 해결할 수 없다면, 저희에게 연락해 주세요. 기꺼이 도와드리겠습니다.

SSH로 접속하려는 팟에 "sleep infinity"와 같은 대기 작업을 지정하지 않은 경우.
실행할 수 없는 명령을 팟에 제공한 경우. 로그를 확인하여 구문 오류가 없는지 점검해 보세요.

✓ **Windows를 실행할 수 있나요?**

현재 RunPod는 Windows를 지원하지 않습니다. 향후 이를 지원할 계획은 있지만, 구체적인 일정은 정해져 있지 않습니다.

✓ **커뮤니티 클라우드에서 신뢰할 수 있는 서버를 어떻게 찾을 수 있나요?**

RunPod는 신뢰할 수 있는 서버를 제공해야 합니다. 모든 나열된 서버는 최소 신뢰성 기준을 충족해야 하며, 대부분 데이터 센터에서 실행됩니다. 그러나 최고의 신뢰성과 보안을 원한다면 보안 클라우드를 사용하는 것이 좋습니다. RunPod는 실시간으로 각 서버와의 연결 상태(heartbeat)를 모니터링하여 서버의 신뢰성을 평가합니다.

✓ **내 팟에 할당된 GPU가 0개인 이유는 무엇인가요?**

이 문제를 피하려면 네트워크 볼륨을 사용하는 것이 가장 좋은 방법입니다. 네트워크 볼륨 사용 방법에서 확인할 수 있습니다.

대부분의 서버에는 물리적 머신당 4~8개의 GPU가 있습니다. 팟을 시작하면 해당 팟은 특정 물리적 머신에 고정됩니다. 팟을 계속 실행(온디맨드)하면 그 GPU는 다른 사용자가 사용할 수 없습니다. 그러나 팟을 중지하면 그 GPU는 다른 사용자에게 할당될 수 있습니다. 다시 팟을

시작하려고 할 때, 특정 머신이 이미 점유되어 있을 수 있습니다. 이 경우, GPU가 할당되지 않은 상태로 팟을 시작할 수 있는 옵션을 제공하여 데이터는 계속 접근할 수 있습니다.

이것은 해당 물리적 머신에 GPU가 없다는 의미일 뿐, 다른 머신에는 여전히 해당 GPU가 있을 수 있습니다. 또한 전송 팟은 계산 능력이 제한적이므로 UI를 통한 파일 전송이 어려울 수 있습니다. 이 경우 터미널 접근이나 클라우드 동기화 방법을 사용해야 할 수도 있습니다.

✓ **네트워크 볼륨이란 무엇인가요?**

네트워크 볼륨은 팟 간에 데이터를 공유할 수 있게 해 주며, 중요한 데이터를 더 유연하게 다룰 수 있게 해 줍니다. 이 기능은 특정 보안 클라우드 데이터 센터에서만 사용할 수 있지만, 점차 더 많은 보안 클라우드 지역으로 확대 중입니다. 네트워크 볼륨을 사용하면 GPU와 함께 데이터를 사용하는 데 있어 파일 전송이 필요한 상황이 드물어질 것입니다.

- **무엇을 해야 할까요? (What if?)**

✓ **내가 자금을 모두 소진하면 어떻게 되나요?**

자금이 부족하여 최소 10분 동안 팟을 실행할 수 없는 경우, 모든 팟은 자동으로 중지됩니다. 팟이 중지되면 컨테이너 디스크 데이터는 손실되지만 볼륨 데이터는 보존됩니다. 적절한 크레딧이 부족하면 팟은 제거될 예정입니다. 크레딧을 추가한 후에는, GPU가 여유 있는 경우 팟을 다시 시작할 수 있습니다.

✓ **내 팟이 실행 중인 머신의 전원이 꺼지면 어떻게 되나요?**

호스트 머신의 전원이 꺼지면, 다시 온라인 상태로 돌아오면 팟이 자동으로 재시작됩니다. 볼륨 데이터는 보존되며, 컨테이너는 처음 실행했을 때와 같은 명령으로 다시 실행됩니다. 하지만 컨테이너 디스크와 메모리 내 데이터는 손실됩니다.

✓ **내 팟이 인터넷 연결을 잃으면 어떻게 되나요?**

호스트 머신은 인터넷 연결이 끊어졌더라도 가능한 한 최선을 다해 팟을 실행합니다. 하지만 인터넷 연결이 필요한 작업이라면 작동하지 않습니다. 호스트가 인터넷 연결을 잃어도 작업이 계속 실행되더라도 요금은 부과되지 않습니다. 물론 호스트가 다시 연결되면 팟을 종료할 수 있습니다.

✓ **내 지출 한도를 초과했다고 나옵니다.**

새로운 계정에는 지출 한도가 설정되어 있으며, 시간이 지남에 따라 점차 증가합니다. 이는 일부 사기꾼이 플랫폼에 방해를 시도하는 것을 방지하기 위해 도입되었습니다. 이 한도가 정상적인 사용에 영향을 미치지 않도록 해야 합니다. 사용 사례를 공유하면 기꺼이 지출 한도를 올려드리겠습니다.

- **법적 사항 (Legal)**

✓ **법적 정보를 볼 수 있나요?**

물론입니다! 법적 페이지를 확인해 주세요.

- **GDPR 준수**

RunPod는 데이터 보호와 개인정보를 중요하게 여기며, GDPR 요구사항을 준수하기 위해 강력한 정책, 절차, 기술적 조치를 구현하고 있습니다.

RunPod는 유럽에서 처리되는 데이터에 대해 GDPR을 준수하나요? 네, RunPod는 유럽 데이터 센터 지역 내에서 처리되는 데이터에 대해 GDPR 요구사항을 완전히 준수하고 있습니다.

✓ **GDPR 준수를 위한 RunPod의 조치**

- 데이터 처리 절차: 개인 데이터를 수집, 저장, 처리, 삭제하는 명확한 절차를 마련하여 투명성과 책임을 보장합니다.
- 데이터 보호 조치: 개인 데이터를 무단 접근, 공개, 변경, 삭제로부터 보호하는 적절한 기술적, 조직적 조치를 취하고 있습니다.
- 동의 메커니즘: GDPR 요구사항에 따라 개인의 데이터를 처리하기 위한 동의를 얻고, 이를 기록합니다. 개인이 동의를 철회할 수 있는 방법도 제공합니다.
- 데이터 주체의 권리: 개인은 데이터에 대한 접근, 수정, 삭제, 처리 제한 등을 요청할 수 있으며, 이를 신속히 처리합니다.
- 데이터 전송 메커니즘: GDPR 요구사항에 맞게, EU 외부로의 개인 데이터 전송을 적법하고 안전하게 처리합니다. GDPR 준수에 대한 문의나 데이터 보호 관행에 대한 궁금한 점이 있으면 support@runpod.io로 문의해 주세요.

- **RunPod에서 결제 카드 거부 관리 방법**

결제 카드가 거부되면 당황스러울 수 있지만 걱정하지 마세요! 문제를 빠르게 해결하고 서비스를 원활하게 계속 이용할 수 있도록 다음 단계를 따라주세요.

✓ **1. 잔액을 미리 충전하기**

결제 문제로 서비스가 중단되는 것을 최소화하려면

- 미리 잔액을 충전하세요. 잔액이 부족하기 전에 며칠 전에 충전하는 것이 좋습니다. 이렇게 하면 문제가 발생해도 충분히 해결할 시간이 생깁니다.
- 자동 충전 설정: 계정 설정에서 잔액 부족 시 자동으로 충전하는 옵션을 활성화할 수 있습니다. 이 기능을 통해 결제 문제가 발생할 가능성을 줄일 수 있습니다.

✓ **2. 카드 발급 은행에 연락하기**

카드가 거부된 경우 가장 먼저 해야 할 일은 다음과 같습니다.

- 카드 발급 은행에 전화하기: RunPod은 결제 거부 사유를 알 수 없으므로, 은행에 직접 연락하여 거부 사유를 확인해야 합니다.

일반적인 거부 이유는 다음과 같습니다.

- ◻ 은행의 사기 방지 시스템이 작동하여 거부된 경우.
- ◻ 국제 거래 제한.
- ◻ 잔액 부족 또는 한도 초과.
- ◻ 은행은 종종 사용자 보호를 위해 거래를 차단하는데, 이를 해결하려면 은행과 상의하는 것이 중요합니다.

- 여러 카드를 시도하지 마세요: 카드를 여러 번 시도하면 계정에 대한 추가 차단이 발생할 수 있습니다. 여러 차례 결제 거부가 발생하면 은행에서 계정에 대한 모든 결제를 차단할 수 있는데, 보통 24시간 후 차단이 해제됩니다. 이 문제를 해결한 후 다시 시도하는 것이 좋습니다.

✓ **3. 카드 거부의 다른 원인**

결제 과정에 영향을 미칠 수 있는 여러 요인이 있습니다:

- 결제 처리자의 리스크 프로파일: RunPod는 Stripe를 결제 처리자로 사용하며, 계정에서 특정 사용 패턴(예: 카드를 자주 변경하거나 이전에 결제 분쟁이 있었던 경우)이 있을 경우 결제가 차단될 수 있습니다.
- 지역 또는 국제 거래 제한: 결제 처리자가 국제 거래를 차단할 수 있습니다.

Stripe에서 지원되는 카드 목록을 확인해 보세요. 카드가 지원되는지 확인할 수 있습니다.

✓ **4. RunPod 지원팀에 문의하기**

만약 은행과 확인 후에도 문제가 해결되지 않는다면, RunPod 지원팀에 연락하여 도움을 받으세요:

지원팀에 문의하기 전, 은행에서 문제를 해결했는지 확인해주세요.
문제가 해결되지 않으면, 저희가 문제 해결을 돕거나 추가적인 안내를 제공하겠습니다.

RunPod에서 사용할 수 있는 GPU 유형

RunPod에서는 다양한 GPU 옵션을 제공합니다. 각 GPU는 메모리 용량에 따라 분류되며, 사용자의 작업에 따라 적합한 선택을 할 수 있습니다. 아래는 RunPod에서 사용 가능한 GPU 목록입니다.

GPU ID	Display Name	Memory (GB)
NVIDIA A100 80GB PCIe	A100 PCIe	80
NVIDIA A100-SXM4-80GB	A100 SXM	80
NVIDIA A30	A30	24
NVIDIA A40	A40	48
NVIDIA H100 NVL	H100 NVL	94
NVIDIA H100 PCIe	H100 PCIe	80
NVIDIA H100 80GB HBM3	H100 SXM	80
NVIDIA L4	L4	24
NVIDIA L40	L40	48
NVIDIA L40S	L40S	48
AMD Instinct MI300X OAM	MI300X	192
NVIDIA RTX 2000 Ada Generation	RTX 2000 Ada	16
NVIDIA GeForce RTX 3070	RTX 3070	8
NVIDIA GeForce RTX 3080	RTX 3080	10
NVIDIA GeForce RTX 3080 Ti	RTX 3080 Ti	12
NVIDIA GeForce RTX 3090	RTX 3090	24
NVIDIA GeForce RTX 3090 Ti	RTX 3090 Ti	24
NVIDIA RTX 4000 Ada Generation	RTX 4000 Ada	20
NVIDIA GeForce RTX 4070 Ti	RTX 4070 Ti	12
NVIDIA GeForce RTX 4080	RTX 4080	16
NVIDIA GeForce RTX 4090	RTX 4090	24
NVIDIA RTX 5000 Ada Generation	RTX 5000 Ada	32
NVIDIA RTX 6000 Ada Generation	RTX 6000 Ada	48
NVIDIA RTX A2000	RTX A2000	6
NVIDIA RTX A4000	RTX A4000	16
NVIDIA RTX A4500	RTX A4500	20
NVIDIA RTX A5000	RTX A5000	24
NVIDIA RTX A6000	RTX A6000	48
Tesla V100-SXM2-16GB	V100 SXM2	16
Tesla V100-SXM2-32GB	V100 SXM2 32GB	32

7. 용어집

◆ **Community Cloud (커뮤니티 클라우드)**

GPU 인스턴스는 개별 컴퓨팅 제공자를 소비자에게 안전하고 신뢰할 수 있는 피어 투 피어 시스템을 통해 연결합니다.

◆ **Datacenter (데이터 센터)**

데이터 센터는 RunPod의 클라우드 컴퓨팅 서비스인 Secure Cloud와 GPU 인스턴스가 호스팅되는 보안된 위치입니다. 이 데이터 센터는 데이터의 안전성과 신뢰성을 보장하기 위해 중복성과 데이터 백업을 갖추고 있습니다.

◆ **Endpoint (엔드포인트)**

엔드포인트는 서버리스 애플리케이션이나 서비스가 액세스될 수 있는 특정 URL을 의미합니다. 이 엔드포인트는 작업을 제출하고 작업 요청의 출력을 검색하는 표준 기능을 제공합니다.

◆ **GPU Instance (GPU 인스턴스)**

GPU 인스턴스는 배포할 수 있는 컨테이너 기반의 GPU 인스턴스입니다. 이 인스턴스는 퍼블릭과 프라이빗 레포지토리 모두에서 몇 초 만에 실행할 수 있습니다. 두 가지 유형으로 제공됩니다.

✓ Secure Cloud

✓ Community Cloud

◆ **Handler (핸들러)**

핸들러는 제출된 입력을 처리하고 결과 출력을 생성하는 기능입니다.

◆ **RunPod (런포드)**

RunPod는 AI 및 머신 러닝 애플리케이션을 위해 설계된 클라우드 컴퓨팅 플랫폼입니다.

◆ **SDKs (소프트웨어 개발 키트)**

RunPod는 플랫폼과 상호 작용할 수 있는 여러 소프트웨어 개발 키트(SDK)를 제공합니다. 이 SDK를 통해 서버리스 기능을 생성하고, 인프라를 관리하며, API와 상호 작용할 수 있습니다.

◆ **Secure Cloud (보안 클라우드)**

T3/T4 데이터 센터에서 실행되는 GPU 인스턴스로, 높은 신뢰성과 보안을 제공합니다.

◆ **Serverless CPU (서버리스 CPU)**

서버리스 CPU는 초단위로 과금되는 서버리스 CPU 컴퓨팅 솔루션입니다. 이는 생산 환경에 자동 확장을 도입하여 애플리케이션의 요구에 따라 동적으로 계산 자원을 조정할 수 있게 설계되었습니다.

◆ **Serverless GPU (서버리스 GPU)**

서버리스 GPU는 초단위로 과금되는 서버리스 GPU 컴퓨팅 솔루션입니다. 이는 생산 환경에 자동 확장을 도입하여 애플리케이션의 요구에 따라 동적으로 GPU 자원을 조정할 수 있게 설계되었습니다.

◆ **Template (템플릿)**

RunPod 템플릿은 Docker 컨테이너 이미지와 구성 파일이 결합된 것입니다.

◆ **Total Workers (총 워커)**

총 워커는 계정에서 사용할 수 있는 워커의 총 수를 나타냅니다. 이는 모든 엔드포인트에서 설정된 최대 워커 수의 합입니다. 총 워커가 부족할 경우, 지원 티켓을 생성하여 문의하시면 됩니다.

◆ **Initial Workers (초기 워커)**

엔드포인트를 처음 생성할 때 RunPod는 워커의 Docker 이미지를 다운로드해야 하며, 이 과정에서 워커는 초기화 중으로 표시됩니다.

◆ **Active (Min) Workers (활성(최소) 워커)**

"항상 켜져 있는" 워커입니다. 활성 워커를 1개 이상 설정하면 워커가 항상 준비 상태로 있어 작업 요청에 즉시 응답할 수 있습니다(콜드 스타트 지연 없음).

✓ 기본값: 0

◆ **Max Workers (최대 워커)**

최대 워커는 엔드포인트에서 동시에 실행할 수 있는 워커 수의 상한을 설정합니다.

✓ 기본값: 3

◆ **Flex Workers (플렉스 워커)**

플렉스 워커는 "가끔 켜져 있는" 워커입니다(최대 워커 - 활성 워커). 이 워커는 트래픽 급증에 대응하기 위해 엔드포인트를 확장하는 데 도움을 줍니다. 플렉스 워커는 작업을 완료하면 "대기 상태"로 전환되어 비용을 절감합니다. 또한, 대기 시간 제한을 조정하여 플렉스 워커가 잠시 더 활성 상태를 유지하도록 할 수 있어 새로운 요청이 올 때 콜드 스타트 지연을 방지할 수 있습니다.

✓ 기본값: 최대 워커(3) - 활성 워커(0) = 3

◆ **Extra Workers (추가 워커)**

RunPod는 워커의 Docker 이미지를 호스트 서버에 캐시하여 더 빠른 확장성을 보장합니다. 트래픽 급증 시 최대 워커 수를 늘리면, 추가 워커가 즉시 플렉스 워커로 추가되어 증가한 수요를 처리합니다.

✓ 기본값: 2

◆ **Throttled Worker (스로틀된 워커)**

선택한 GPU가 워커 시작 시 사용할 수 없는 경우, 리소스가 사용 가능해질 때까지 워커가 스로틀됩니다.

◆ **Stale Workers (구식 워커)**

엔드포인트 구성을 변경하거나 워커를 새로운 Docker 이미지로 업데이트할 때, 워커는 구식으로 표시되고 현재 작업을 마친 후 새로운 워커로 교체됩니다. 이 교체는 롤링 업데이트 방식으로 진행되며, 엔드포인트의 최대 워커 수에 따라 일부 구식 워커가 새 워커로 교체됩니다.

런팟(Runpod) 사용자 가이드

발 행 일 2024년 12월 2일 초판
발 행 처 플레이데이터평생교육원
주 소 지 서울시 금천구 가산디지털1로 25 대륭테크노타운 17차 18층
문 의 처 02-754-7303