

API Pricing Recalculate

Dataiku API Service: pricing

Created in API designer menu in dataiku project.

Endpoint: Recalculate

Background

The Pricing backend initially finds similarities between CPCs and from the 10 closest or so, compute a recommended price for the target CPC (median + business rules, overlays, etc.)

However some of the initial neighbors CPCs might not be comparable from business point of view and they remove them. The new recommended price must be recomputed, that is where the API is used.

Goal

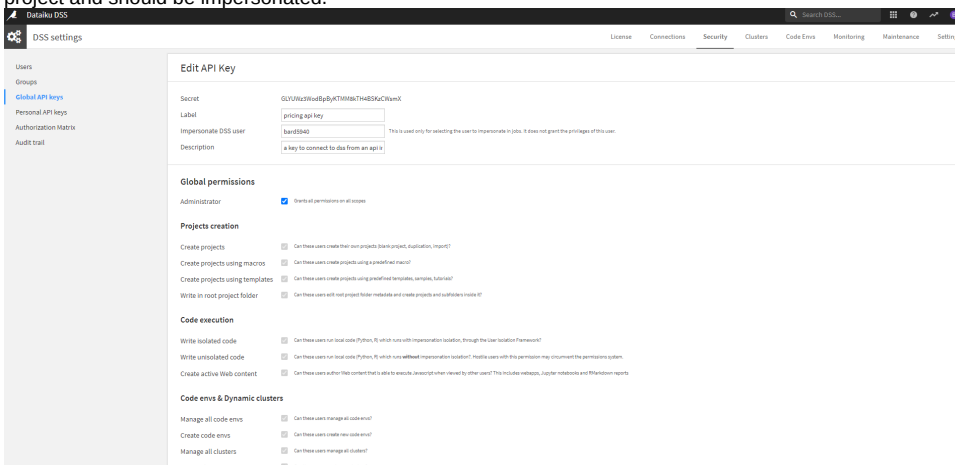
Recalculate the recommended price given the target and a list of neighbor CPCs.

Requirements

- Code shared between API and project, not duplicated
- Access to project's intermediate datasets to re compute only what is necessary
- Dataiku project API Webapp connections in different environment should match

Solution

- Code shared between project initial computation and API is in functions in project's library, not directly in recipes
- Datasets: a mix of PostgreSQL tables and managed folders in project, so we choose to connect to project's and access datasets with dataiku api (alternative would have been direct connection to postgresql if only postgresql tables for example, etc)
- API connects to remote DSS instance using a "Global API key" that should be set at instance level. This key must have appropriate rights for project and should be impersonated.



- Remote DSS is set depending on API host

```
18 # Host configuration
19 # hostname is something like: dku-mad-pricing-on-dss-test-standard-1-855fdd5784-kqvcx
20 hostname = socket.gethostname()
21 if hostname == "ce-bda-dss-design-ew1-prod": # PROD DESIGN
22     dataiku.set_remote_dss("http://ce-bda-dss-design-ew1-prod:10000", "GLYUwz3wodBpYkTM8kTH4BSKzCwsmx")
23     os.environ["DKU_CURRENT_PROJECT_KEY"] = 'PBE_MASTER'
24     version = get_config_version()
25 elif "dss-prod-standard" in hostname: # PROD AUTOMATION
26     dataiku.set_remote_dss("http://ce-bda-dss-automation-ew1-prod:10000", "3rmX9t90tkOcFlopBAco5jMrpQRhrwsN")
27     os.environ["DKU_CURRENT_PROJECT_KEY"] = 'PBE_MASTER'
28     version = get_config_version()
29 elif hostname == "ce-bda-dss-design-ew1-test": # TEST DESIGN
30     dataiku.set_remote_dss("http://ce-bda-dss-design-ew1-test:10000", "ytYjYLaz7j9Nrh1Q66yEkT9DjGQCzqo7")
31     os.environ["DKU_CURRENT_PROJECT_KEY"] = 'PBE_MASTER'
32     version = get_config_version()
33 elif "dss-test-standard" in hostname: # TEST AUTOMATION
34     dataiku.set_remote_dss("http://ce-bda-dss-automation-ew1-test:10000", "1zFCkW5vc189j4QN19UPDKxhs1sOD0qP")
35     os.environ["DKU_CURRENT_PROJECT_KEY"] = 'PBE_MASTER'
36     version = get_config_version()
37 else:
38     class ConfigError(Exception):
39         pass
40     raise ConfigError(f"Hostname '{hostname}' is not mapped to a DSS config.")
```

Additional settings

- **Security:** The API service is not public but an API key is created for webapp that should use it when calling the API. If other components need to use the API they should use another API key.

Deployment

Actually done by data ops team.

Since API host (hostname) is important, make sure deployment follows needed naming convention (for prod, test env).

Additionally we deploy on a Kubernetes cluster with a load balancer, some settings are needed.

Steps:

1. In API Designer, Create a Version and Publish it on deployer
2. In Local Deployer, API Services, find the correct service and version, and deploy it: new deployment or update existing deployment.
3. Before the actual deployment, check the settings

The image shows two screenshots of the API Designer interface, specifically the 'Settings' page for a deployment named 'pricing-on-dss-prod-standard'.

Top Screenshot: Replication settings

- Replication settings:** These settings control the scalability of the underlying Kubernetes deployments.
- Override infrastructure settings:** ☒ If unchecked, settings defined at the infrastructure level are used.
- Replica number:** 1 (Number of pod replicas in Kubernetes)
- Autoscaler:** ☒ Enable Kubernetes Horizontal Pod Autoscaler
- API Node server sizing and tuning:** These settings control internal sizing parameters of the API node server.
- Override infrastructure settings:** ☐ If unchecked, settings defined at the infrastructure level are used.
- Container limits:** These settings control CPU and memory limits that are set on each container. It is recommended to be familiar with Kubernetes "request" and "limit" parameters to use this.
- Override infrastructure settings:** ☐ If unchecked, settings defined at the infrastructure level are used.

Bottom Screenshot: Service exposition

- Service exposition:** These settings control exposition of service.
- Override infrastructure settings:** ☒ If unchecked, settings defined at the infrastructure level are used.
- Exposition mode:** Load balancer
- Port:** 80
- Service annotations:** networking.gke.io/load-bal- Internal
- ADD A KEY/VALUE PAIR**
- Raw text edit**
- Internal:** ☒ If available in the cloud provider. Dependent on the value of the "cloud.provider" property on the container config.
- Force IP:** (If available in the cloud provider)
- Restrict to source IPs:** (Type a value and press enter to add it to the list. If available in the cloud provider.)