Craft Providers

Release 1.0.0

Canonical Ltd.

Jul 14, 2021

PUBLIC APIS:

Here you will find all of the provider documentation…

CHAPTER

ONE

EXECUTORS

1.1 Abstract Executor

class craft_providers.**Executor**

Bases: abc.ABC

Interfaces to execute commands and move data in/out of an environment.

abstract execute_popen(*command*, *env=None*, ***kwargs*)

Execute a command in instance, using subprocess.Popen().

The process' environment will inherit the execution environment's default environment (PATH, etc.), but can be additionally configured via env parameter.

Parameters

- **command** (List[str]) – Command to execute.
- **env** (Optional[Dict[str, Optional[str]]]) – Additional environment to set for process.
- **kwargs** – Additional keyword arguments to pass.

Return type Popen

Returns Popen instance.

abstract execute_run(*command*, *env=None*, ***kwargs*)

Execute a command using subprocess.run().

The process' environment will inherit the execution environment's default environment (PATH, etc.), but can be additionally configured via env parameter.

Parameters

- **command** (List[str]) – Command to execute.
- **env** (Optional[Dict[str, Optional[str]]]) – Additional environment to set for process.
- **kwargs** – Keyword args to pass to subprocess.run().

Return type CompletedProcess

Returns Completed process.

Raises **subprocess.CalledProcessError** – if command fails and check is True.

abstract pull_file(***, *source*, *destination*)

Copy a file from the environment to host.

Parameters

- **source** (Path) – Environment file to copy.
- **destination** (Path) – Host file path to copy to. Parent directory (destination.parent) must exist.

Raises

- **FileNotFoundError** – If source file or destination's parent directory does not exist.
- **ProviderError** – On error copying file.

Return type None

abstract push_file(*, *source*, *destination*)

Copy a file from the host into the environment.

Parameters

- **source** (Path) – Host file to copy.
- **destination** (Path) – Target environment file path to copy to. Parent directory (destination.parent) must exist.

Raises

- **FileNotFoundError** – If source file or destination's parent directory does not exist.
- **ProviderError** – On error copying file.

Return type None

abstract push_file_io(*, *destination*, *content*, *file_mode*, *group='root'*, *user='root'*)

Create a file with specified content and file mode.

Parameters

- **destination** (Path) – Path to file.
- **content** (BytesIO) – Contents of file.
- **file_mode** (str) – File mode string (e.g. '0644').
- **group** (str) – File owner group.
- **user** (str) – File owner user.

Return type None

1.2 LXD Executor

class craft_providers.lxd.**LXDInstance**(*, *name*, *default_command_environment=None*, *project='default'*, *remote='local'*, *lxc=None*)

Bases: *craft_providers.executor.Executor*

LXD Instance Lifecycle.

Parameters

- **name** (str) –
- **default_command_environment** (Optional[Dict[str, Optional[str]]]) –
- **project** (str) –
- **remote** (str) –

- **lxc** (Optional[LXC]) –

delete(*force=True*)
Delete instance.

Parameters **force** (bool) – Delete even if running.

Raises **LXDError** – On unexpected error.

Return type None

execute_popen(*command*, *env=None*, ***kwargs*)
Execute a command in instance, using subprocess.Popen().

The process' environment will inherit the execution environment's default environment (PATH, etc.), but can be additionally configured via env parameter.

Parameters

- **command** (List[str]) – Command to execute.
- **env** (Optional[Dict[str, Optional[str]]]) – Additional environment to set for process.
- **kwargs** – Additional keyword arguments to pass.

Return type Popen

Returns Popen instance.

execute_run(*command*, *env=None*, ***kwargs*)
Execute a command using subprocess.run().

The process' environment will inherit the execution environment's default environment (PATH, etc.), but can be additionally configured via env parameter.

Parameters

- **command** (List[str]) – Command to execute.
- **env** (Optional[Dict[str, Optional[str]]]) – Additional environment to set for process.
- **kwargs** – Keyword args to pass to subprocess.run().

Return type CompletedProcess

Returns Completed process.

Raises **subprocess.CalledProcessError** – if command fails and check is True.

exists()
Check if instance exists.

Return type bool

Returns True if instance exists.

Raises **LXDError** – On unexpected error.

is_mounted(***, *host_source*, *target*)
Check if path is mounted at target.

Parameters

- **host_source** (Path) – Host path to check.
- **target** (Path) – Instance path to check.

Return type bool

Returns True if host_source is mounted at target.

Raises **LXDError** – On unexpected error.

is_running()
Check if instance is running.

Return type bool

Returns True if instance is running.

Raises **LXDError** – On unexpected error.

launch(*, *image*, *image_remote*, *map_user_uid=False*, *ephemeral=False*)
Launch instance.

Parameters

- **image** (str) – Image name to launch.
- **image_remote** (str) – Image remote name.
- **uid** – Host user ID to map to instance root.
- **ephemeral** (bool) – Flag to enable ephemeral instance.
- **map_user_uid** (bool) –

Raises **LXDError** – On unexpected error.

Return type None

mount(*, *host_source*, *target*, *device_name=None*)
Mount host source directory to target mount point.

Checks first to see if already mounted. If no device name is given, it will be generated with the format "disk-{target.as_posix()}".

Parameters

- **host_source** (Path) – Host path to mount.
- **target** (Path) – Instance path to mount to.
- **device_name** (Optional[str]) – Name for disk device.

Raises **LXDError** – On unexpected error.

Return type None

pull_file(*, *source*, *destination*)
Copy a file from the environment to host.

Parameters

- **source** (Path) – Environment file to copy.
- **destination** (Path) – Host file path to copy to. Parent directory (destination.parent) must exist.

Raises

- **FileNotFoundError** – If source file or destination's parent directory does not exist.
- **LXDError** – On unexpected error copying file.

Return type None

push_file(*, *source*, *destination*)
Copy a file from the host into the environment.

Parameters

- **source** (Path) – Host file to copy.
- **destination** (Path) – Target environment file path to copy to. Parent directory (destination.parent) must exist.

Raises

- **FileNotFoundError** – If source file or destination's parent directory does not exist.
- **LXDError** – On unexpected error copying file.

Return type None

push_file_io(*, *destination*, *content*, *file_mode*, *group='root'*, *user='root'*)
Create file with content and file mode.

Parameters

- **destination** (Path) – Path to file.
- **content** (BytesIO) – Contents of file.
- **file_mode** (str) – File mode string (e.g. '0644').
- **group** (str) – File group owner/id.
- **user** (str) – File user owner/id.

Raises **LXDError** – On unexpected error.

Return type None

start()
Start instance.

Raises **LXDError** – on unexpected error.

Return type None

stop()
Stop instance.

Raises **LXDError** – on unexpected error.

Return type None

supports_mount()
Check if instance supports mounting from host.

Return type bool

Returns True if mount is supported.

unmount(*target*)
Unmount mount target shared with host.

Parameters **target** (Path) – Target shared with host to unmount.

Raises **LXDError** – On failure to unmount target.

Return type None

unmount_all()
Unmount all mounts shared with host.

Raises **LXDError** – On failure to unmount target.

Return type None

1.3 Multipass Executor

class craft_providers.multipass.**MultipassInstance**(*, *name*, *multipass=None*)
Bases: *craft_providers.executor.Executor*

Multipass Instance Lifecycle.

Parameters

- **name** (str) – Name of multipass instance.
- **multipass** (Optional[Multipass]) –

delete()
Delete instance and purge.

Return type None

execute_popen(*command*, *env=None*, ***kwargs*)
Execute process in instance using subprocess.Popen().

The process' environment will inherit the execution environment's default environment (PATH, etc.), but can be additionally configured via env parameter.

Parameters

- **command** (List[str]) – Command to execute.
- **env** (Optional[Dict[str, Optional[str]]]) – Additional environment to set for process.
- **kwargs** – Additional keyword arguments for subprocess.Popen().

Return type Popen

Returns Popen instance.

execute_run(*command*, *env=None*, ***kwargs*)
Execute command using subprocess.run().

The process' environment will inherit the execution environment's default environment (PATH, etc.), but can be additionally configured via env parameter.

Parameters

- **command** (List[str]) – Command to execute.
- **env** (Optional[Dict[str, Optional[str]]]) – Additional environment to set for process.
- **kwargs** – Keyword args to pass to subprocess.run().

Return type CompletedProcess

Returns Completed process.

Raises **subprocess.CalledProcessError** – if command fails and check is True.

exists()
Check if instance exists.

Return type bool

Returns True if instance exists.

Raises **MultipassError** – On unexpected failure.

is_mounted(*, *host_source*, *target*)
Check if path is mounted at target.

Parameters

- **host_source** (Path) – Host path to check.
- **target** (Path) – Instance path to check.

Return type bool

Returns True if host_source is mounted at target.

Raises **MultipassError** – On unexpected failure.

is_running()
Check if instance is running.

Return type bool

Returns True if instance is running.

Raises **MultipassError** – On unexpected failure.

launch(*, *image*, *cpus=2*, *disk_gb=256*, *mem_gb=2*)
Launch instance.

Parameters

- **image** (str) – Name of image to create the instance with.
- **instance_cpus** – Number of CPUs.
- **instance_disk_gb** – Disk allocation in gigabytes.
- **instance_mem_gb** – Memory allocation in gigabytes.
- **instance_name** – Name of instance to use/create.
- **instance_stop_time_mins** – Stop time delay in minutes.
- **cpus** (int) –
- **disk_gb** (int) –
- **mem_gb** (int) –

Raises **MultipassError** – On unexpected failure.

Return type None

mount(*, *host_source*, *target*)
Mount host host_source directory to target mount point.

Checks first to see if already mounted.

Parameters

- **host_source** (Path) – Host path to mount.
- **target** (Path) – Instance path to mount to.

Raises **MultipassError** – On unexpected failure.

Return type None

pull_file(*, *source*, *destination*)
Copy a file from the environment to host.

Parameters

- **source** (Path) – Environment file to copy.
- **destination** (Path) – Host file path to copy to. Parent directory (destination.parent) must exist.

Raises

- **FileNotFoundError** – If source file or destination's parent directory does not exist.
- **MultipassError** – On unexpected error copying file.

Return type None

push_file(*, *source*, *destination*)

Copy a file from the host into the environment.

Parameters

- **source** (Path) – Host file to copy.
- **destination** (Path) – Target environment file path to copy to. Parent directory (destination.parent) must exist.

Raises

- **FileNotFoundError** – If source file or destination's parent directory does not exist.
- **MultipassError** – On unexpected error copying file.

Return type None

push_file_io(*, *destination*, *content*, *file_mode*, *group='root'*, *user='root'*)

Create file with content and file mode.

Multipass transfers data as "ubuntu" user, forcing us to first copy a file to a temporary location before moving to a (possibly) root-owned location and with appropriate permissions.

Parameters

- **destination** (Path) – Path to file.
- **content** (BytesIO) – Contents of file.
- **file_mode** (str) – File mode string (e.g. '0644').
- **group** (str) – File group owner/id.
- **user** (str) – File user owner/id.

Return type None

start()

Start instance.

Raises **MultipassError** – On unexpected failure.

Return type None

stop(*, *delay_mins=0*)

Stop instance.

Parameters **delay_mins** (int) – Delay shutdown for specified minutes.

Raises **MultipassError** – On unexpected failure.

Return type None

unmount(*target*)
Unmount mount target shared with host.

Parameters **target** (Path) – Target shared with host to unmount.

Raises **MultipassError** – On failure to unmount target.

Return type None

unmount_all()
Unmount all mounts shared with host.

Raises **MultipassError** – On failure to unmount target.

Return type None

CHAPTER

TWO

BASES

2.1 Abstract Base

class craft_providers.**Base**

Bases: abc.ABC

Interface for providers to configure instantiated environments.

Defines how to setup/configure an environment that has been instantiated by a provider and prepare it for some operation, e.g. execute build. It must account for:

(1) the OS type and version.

(2) the provided image that was launched, e.g. bootstrapping a minimal image versus a more fully featured one.

(3) any dependencies that are required for the operation to complete, e.g. installed applications, networking configuration, etc. This includes any environment configuration that the application will assume is available.

Variables **compatibility_tag** – Tag/Version for variant of build configuration and setup. Any change to this version would indicate that prior [versioned] instances are incompatible and must be cleaned. As such, any new value should be unique to old values (e.g. incrementing). It is suggested to extend this tag, not overwrite it, e.g.: compatibilty_tag = f"{appname}-{Base.compatibility_tag}.{apprevision}" to ensure base compatibility levels are maintained.

abstract **get_command_environment**()

Get command environment to use when executing commands.

Return type Dict[str, Optional[str]]

Returns Dictionary of environment, allowing None as a value to indicate that a value should be unset.

abstract **setup**(*, *executor*, *retry_wait=0.25*, *timeout=None*)

Prepare base instance for use by the application.

Wait for environment to become ready and configure it. At completion of setup, the executor environment should have networking up and have all of the installed dependencies required for subsequent use by the application.

Setup may be called more than once in a given instance to refresh/update the environment.

If timeout is specified, abort operation if time has been exceeded.

Parameters

- **executor** (*Executor*) – Executor for target container.
- **retry_wait** (float) – Duration to sleep() between status checks (if required).

- **timeout** (Optional[float]) – Timeout in seconds.

Raises **ProviderError** – on timeout or unexpected error.

Return type None

abstract wait_until_ready(*, *executor*, *retry_wait=0.25*, *timeout=None*)

Wait until base instance is ready.

Ensure minimum-required boot services are running. This would be used when starting an environment's container/VM after already [recently] running setup(), e.g. rebooting the instance. Allows the environment to be used without the cost incurred by re-executing the steps unnecessarily.

If timeout is specified, abort operation if time has been exceeded.

Parameters

- **executor** (*Executor*) – Executor for target container.
- **retry_wait** (float) – Duration to sleep() between status checks (if required).
- **timeout** (Optional[float]) – Timeout in seconds.

Raises **ProviderError** – on timeout or unexpected error.

Return type None

2.2 Buildd Base

class craft_providers.bases.**BuilddBase**(*, *alias*, *environment=None*, *hostname='craft-buildd-instance'*)

Bases: *craft_providers.base.Base*

Support for Ubuntu minimal buildd images.

Variables

- **compatibility_tag** – Tag/Version for variant of build configuration and setup. Any change to this version would indicate that prior [versioned] instances are incompatible and must be cleaned. As such, any new value should be unique to old values (e.g. incrementing). It is suggested to extend this tag, not overwrite it, e.g.: compatibilty_tag = f"{appname}-{BuildBase.compatibility_tag}.{apprevision}" to ensure base compatibility levels are maintained.
- **instance_config_path** – Path to persistent environment configuration used for compatibility checks (or other data). Set to /etc/craft-instance.conf, but may be overridden for application-specific reasons.
- **instance_config_class** – Class defining instance configuration. May be overridden with an application-specific subclass of InstanceConfiguration to enable application-specific extensions.

Parameters

- **alias** (BuilddBaseAlias) – Base alias / version.
- **environment** (Optional[Dict[str, Optional[str]]]) – Environment to set in /etc/environment.
- **hostname** (str) – Hostname to configure.

get_command_environment()

Get command environment to use when executing commands.

Return type Dict[str, Optional[str]]

Returns Dictionary of environment, allowing None as a value to indicate that a value should be unset.

instance_config_class
alias of craft_providers.bases.instance_config.InstanceConfiguration

setup(*, *executor*, *retry_wait=0.25*, *timeout=None*)
Prepare base instance for use by the application.

Wait for environment to become ready and configure it. At completion of setup, the executor environment should have networking up and have all of the installed dependencies required for subsequent use by the application.

Setup may be called more than once in a given instance to refresh/update the environment.

If timeout is specified, abort operation if time has been exceeded.

Guarantees provided by this setup:

- configured /etc/environment
- configured hostname
- networking available (IP & DNS resolution)
- apt cache up-to-date
- snapd configured and ready
- system services are started and ready

Parameters

- **executor** (*Executor*) – Executor for target container.
- **retry_wait** (float) – Duration to sleep() between status checks (if required).
- **timeout** (Optional[float]) – Timeout in seconds.

Raises

- **BaseCompatibilityError** – if instance is incompatible.
- **BaseConfigurationError** – on other unexpected error.

Return type None

wait_until_ready(*, *executor*, *retry_wait=0.25*, *timeout=None*)
Wait until base instance is ready.

Ensure minimum-required boot services are running. This would be used when starting an environment's container/VM after already [recently] running setup(), e.g. rebooting the instance. Allows the environment to be used without the cost incurred by re-executing the steps unnecessarily.

If timeout is specified, abort operation if time has been exceeded.

Guarantees provided by this wait:

- networking available (IP & DNS resolution)
- system services are started and ready

Parameters

- **executor** (*Executor*) – Executor for target container.

- **retry_wait** (float) – Duration to sleep() between status checks (if required).
- **timeout** (Optional[float]) – Timeout in seconds.

Raises **ProviderError** – on timeout or unexpected error.

Return type None

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX