

# Building Custom Images

The ability to create custom [Images](#) is a powerful feature of the Kasm framework. Administrators may choose to maintain and automatically deploy always up-to-date Images to the Kasm deployment with no user downtime. See [Image Maintenance Process](#) for more details.

Administrators will create Docker Images that import from an existing [Default or Core Image](#) published by the Kasm Technologies team. The **Core** images contain the minimal set of configurations that is necessary for the images to work within the platform.

The `kasmweb/core-ubuntu-bionic` image is the preferred core image and is based on Ubuntu 18.04 LTS. Generally speaking most programs that can be installed on Ubuntu can be installed inside this image. The exception are programs that come delivered as containers themselves such as [Snaps](#) and some [FlatPaks](#). Administrators will need working knowledge of Docker and scripting to add custom software and configurations to the Docker images.

A Git repository with several example Dockerfiles that demonstrate how to build full desktop or single application images is available on [GitHub](#).

In this guide we will walk through the basic steps for creating a custom Image. These steps can be conducted on the server with the Kasm [Agent](#) installed.

## ❗ Important

The Kasm team publishes new editions of the core image at every release cycle. Be sure to build your image based on the appropriate tag for your deployed Kasm version.

```
FROM kasmweb/core-ubuntu-bionic:1.10.0
```

## ❗ Note

It is also possible to base a custom Image on any of the default Images published. See [Default Images](#) for a list.

## Basic Build



## Tip

If performing a build directly on the Kasm Workspaces server or Agent, disable **Automatically Prune Images** for the applicable Agent. See [Agent Settings](#) for more details.

1. Below is the baseline `Dockerfile` used to create a custom Kasm Image. There is a pre-defined sections where customizations are to be added. The statements before and after this section should not be modified. In this example a single customization is added : an file named **hello.txt** is created on the desktop.

Create a file named `Dockerfile` with the following contents.

```
FROM kasmweb/core-ubuntu-bionic:1.10.0
USER root

ENV HOME /home/kasm-default-profile
ENV STARTUPDIR /dockerstartup
ENV INST_SCRIPTS $STARTUPDIR/install
WORKDIR $HOME

##### Customize Container Here #####

RUN touch $HOME/Desktop/hello.txt

##### End Customizations #####

RUN chown 1000:0 $HOME
RUN $STARTUPDIR/set_user_permission.sh $HOME


ENV HOME /home/kasm-user
WORKDIR $HOME
RUN mkdir -p $HOME && chown -R 1000:0 $HOME

USER 1000
```

2. Build the Image.

```
sudo docker build -t sublime-text:example -f Dockerfile .
```

3. Log into the Kasm UI as an administrator and register a new Image by selecting **Images** -> **Create New Images**

 **Update Image**

**Name\***

**Description\***

**Friendly Name\***

**Thumbnail URL**

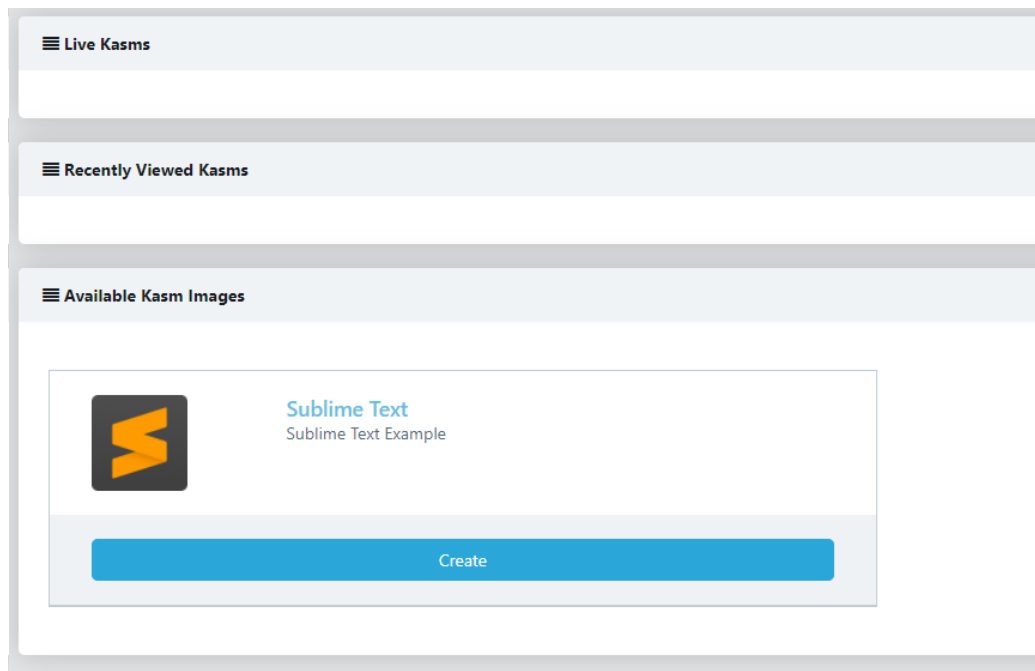
**Cores\***

**Memory (MB)\***

**Enabled**  
☒

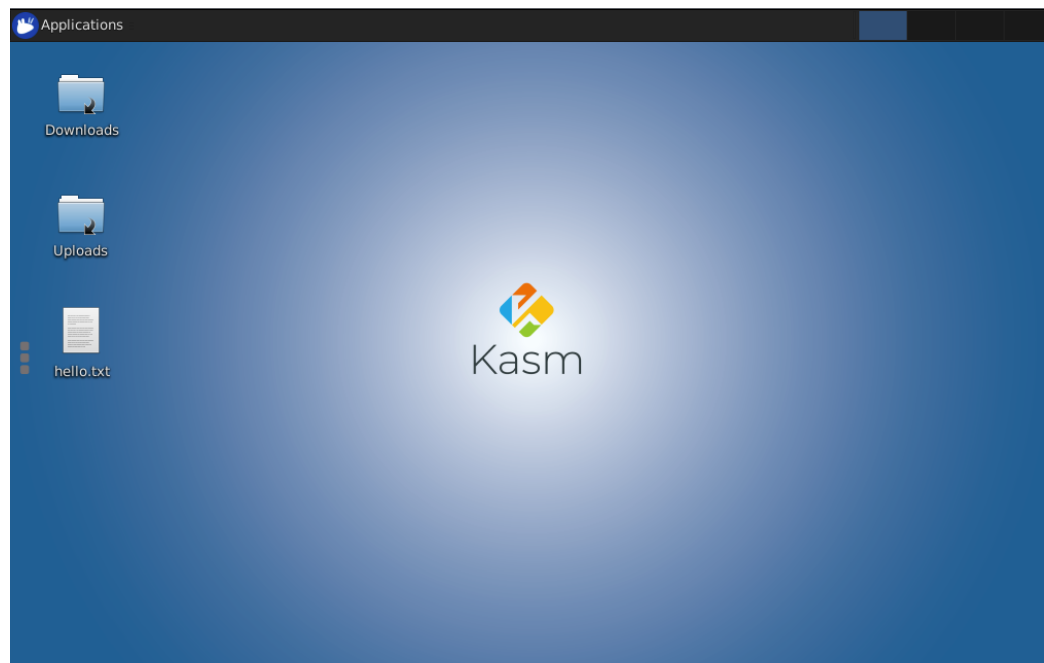
### Register New Image

4. From the Kasm Dashboard click **Create** next the the Sublime Text Image.



### New Image Available

5. A new session is created with the **hello.txt** file we created.



*Running the custom Image*

## Installing Software

1. Now it's time to do something more useful. We will update the `Dockerfile` to actually install Sublime Text.

Update the `Dockerfile` with the following contents.

```
FROM kasmweb/core-ubuntu-bionic:1.10.0
USER root

ENV HOME /home/kasm-default-profile
ENV STARTUPDIR /dockerstartup
ENV INST_SCRIPTS $STARTUPDIR/install
WORKDIR $HOME

##### Customize Container Here #####

RUN wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | apt-key add - \
    && apt-get update \
    && apt-get install -y apt-transport-https \
    && echo "deb https://download.sublimetext.com/ apt/stable/" | tee /etc/apt/sources.
    && apt-get update \
    && apt-get install sublime-text \
    && cp /usr/share/applications/sublime_text.desktop $HOME/Desktop/ \
    && chmod +x $HOME/Desktop/sublime_text.desktop \
    && chown 1000:1000 $HOME/Desktop/sublime_text.desktop

##### End Customizations #####

RUN chown 1000:0 $HOME
RUN $STARTUPDIR/set_user_permission.sh $HOME

ENV HOME /home/kasm-user
WORKDIR $HOME
RUN mkdir -p $HOME && chown -R 1000:0 $HOME

USER 1000
```

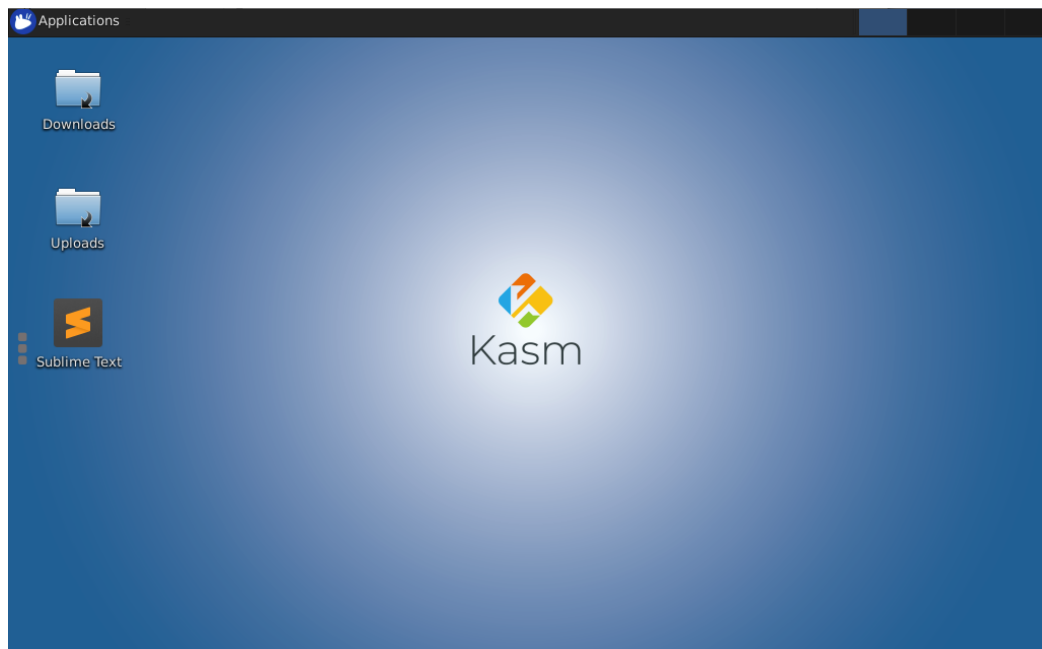
### ❗ Note

Creating desktop icons is a common need. This example highlights how applications , when installed often place a `.desktop` file in `/usr/share/applications` . This file can often be copied without modification to the desktop `$HOME/Desktop/` . It is important to mark the file as executable and ensure the ownership is changed to user and group `1000`

### Reference:

- [https://www.sublimetext.com/docs/3/linux\\_repositories.html](https://www.sublimetext.com/docs/3/linux_repositories.html)

2. Rebuild the Image and create a new session and verify Sublime Text is installed and an icon is present on the desktop.



*Sublime Text is Installed*

## Custom Startup

1. Next we will utilize the `custom_startup.sh` interface point to launch Sublime text when the session starts. Create the script and mark it executable. This script will run and the standard user ( `1000` ) context when the session starts. Utilize the built-in command `/usr/bin` `/desktop_ready` to ensure Sublime Text starts after the Kasm desktop environment.

Update the `Dockerfile` with the following contents.

```
FROM kasmweb/core-ubuntu-bionic:1.10.0
USER root

ENV HOME /home/kasm-default-profile
ENV STARTUPDIR /dockerstartup
ENV INST_SCRIPTS $STARTUPDIR/install
WORKDIR $HOME

##### Customize Container Here #####

RUN wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | apt-key add - \
  && apt-get update \
  && apt-get install -y apt-transport-https \
  && echo "deb https://download.sublimetext.com/ apt/stable/" | tee /etc/apt/sources.
  && apt-get update \
  && apt-get install sublime-text \
  && cp /usr/share/applications/sublime_text.desktop $HOME/Desktop/ \
  && chmod +x $HOME/Desktop/sublime_text.desktop \
  && chown 1000:1000 $HOME/Desktop/sublime_text.desktop

RUN echo "/usr/bin/desktop_ready && /opt/sublime_text/sublime_text &" > $STARTUPDIR/custom
&& chmod +x $STARTUPDIR/custom_startup.sh

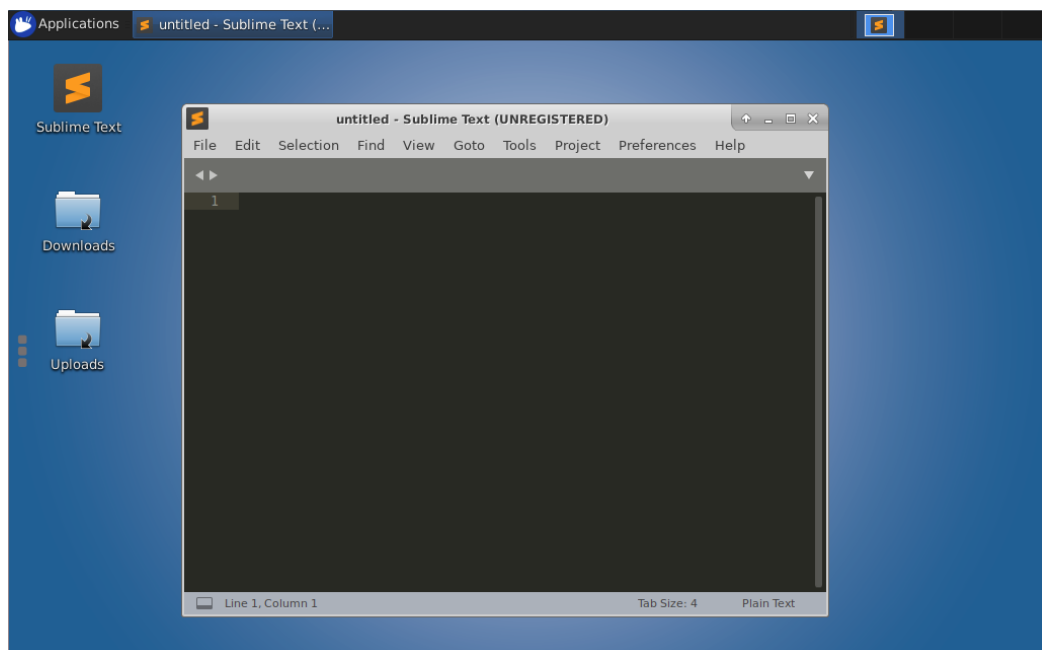
##### End Customizations #####

RUN chown 1000:0 $HOME
RUN $STARTUPDIR/set_user_permission.sh $HOME

ENV HOME /home/kasm-user
WORKDIR $HOME
RUN mkdir -p $HOME && chown -R 1000:0 $HOME

USER 1000
```

2. Rebuild the Image and create a new session. Sublime Text should start automatically.



**Sublime Text Started Automatically**

# Desktop Background

1. The background can be changed by overwriting the `/usr/share/extra/backgrounds/bg_default.png` file.

Update the `Dockerfile` with the following contents.

```
FROM kasmweb/core-ubuntu-bionic:1.10.0
USER root

ENV HOME /home/kasm-default-profile
ENV STARTUPDIR /dockerstartup
ENV INST_SCRIPTS $STARTUPDIR/install
WORKDIR $HOME

##### Customize Container Here #####

RUN wget -q0 - https://download.sublimetext.com/sublimehq-pub.gpg | apt-key add - \
    && apt-get update \
    && apt-get install -y apt-transport-https \
    && echo "deb https://download.sublimetext.com/ apt/stable/" | tee /etc/apt/sources.
    && apt-get update \
    && apt-get install sublime-text \
    && cp /usr/share/applications/sublime_text.desktop $HOME/Desktop/ \
    && chmod +x $HOME/Desktop/sublime_text.desktop \
    && chown 1000:1000 $HOME/Desktop/sublime_text.desktop

RUN echo "/usr/bin/desktop_ready && /opt/sublime_text/sublime_text &" > $STARTUPDIR/custom
&& chmod +x $STARTUPDIR/custom_startup.sh

RUN wget https://cdn.hipwallpaper.com/i/92/9/0Ts6mr.png -O /usr/share/extra/backgrounds/l

##### End Customizations #####

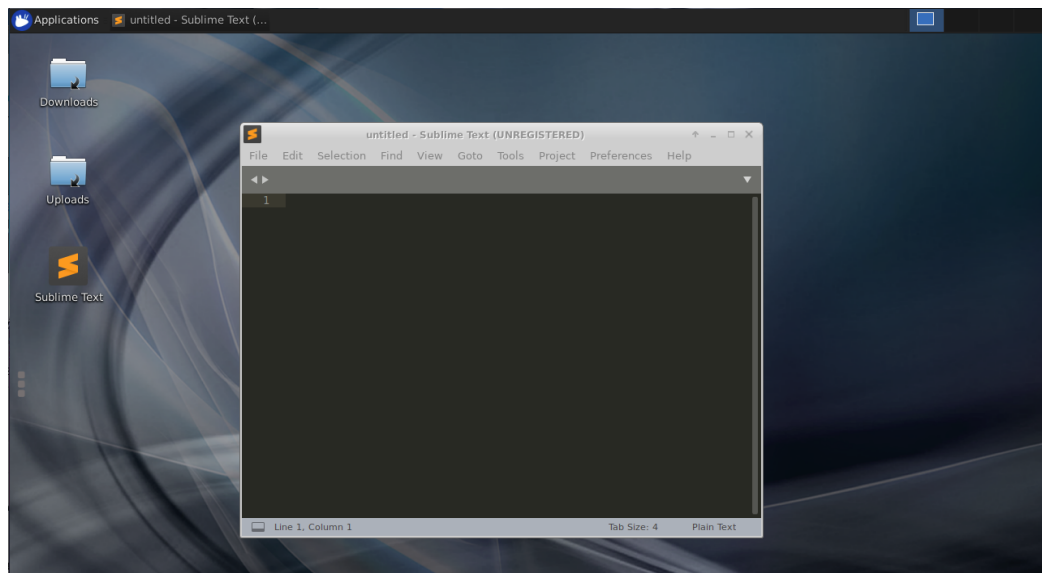
RUN chown 1000:0 $HOME
RUN $STARTUPDIR/set_user_permission.sh $HOME

ENV HOME /home/kasm-user
WORKDIR $HOME
RUN mkdir -p $HOME && chown -R 1000:0 $HOME

USER 1000
```

2. Rebuild the image and create a new session. Sublime Text should start automatically.





*Custom Desktop Background*

## Push to Registry

To simplify Image management, it is recommended to utilize a docker container registry. This example will utilize a container registry that is provided by GitLab.

1. Login to the docker container registry. You will be prompted for a username and password. GitLab provides the ability to create [Personal Access Tokens](#) with permissions that are limited to read and/or write permissions to the registry.

```
sudo docker login registry.gitlab.com
```


2. Build the image, using the registry location as part of the image name.

```
sudo docker build -t registry.gitlab.com/my-company/my-project/sublime-text:example -f Dockerfile .
```

3. Push the image to the registry

```
sudo docker push registry.gitlab.com/my-company/my-project/sublime-text:example
```

4. Register the image via the Kasm UI. Enter the full image name and tag as well as the docker registry. Enter the gitlab username in the **Docker Registry Username** field and the Gitlab password or access token under the **Docker Registry Password** field

 **Update Image**

**Name\***

**Description\***

**Friendly Name\***

**Thumbnail URL**

**Cores\***

**Memory (MB)\***

**Enabled**  
☒

**Docker Registry**

**Docker Registry Username**

**Docker Registry Password**

### *Using a Docker Container Registry*

## General Docker Images

Kasm Workspaces is intended primarily for UI streaming containers, however, Workspaces can also orchestrate containers using any image. Images that are not based on one of the Kasm maintained [core images](#) will be incompatible with some features.

- Web Filtering
- URL Categorization
- Connecting to container through the UI

By default, Workspaces applies a restart policy to containers of 'unless-stopped', which means the container is automatically restarted unless it is manually stopped. This may or may not be desired. To launch containers that performed a task and exited, the default restart policy must be overridden. In the Workspaces Admin UI, navigate to Images, edit the desired image and in the **Docker Run Config Override (JSON)** field, set the restart policy to 'on-failure' as shown here.

```
{"restart_policy":{"Name":"on-failure","MaximumRetryCount":5}}
```

With this policy applied, when a user creates an instance of this image, it will be automatically removed when the container is finished running.