```
node {
    checkout scm
    def customImage = docker.build("my-image:${env.BUILD_ID}")
    customImage.push()

    customImage.push('latest')
}
```

Using a remote Docker server

By default, the plugin:docker-workflow[Docker Pipeline] plugin will communicate with a local Docker daemon, typically accessed through `/var/run/docker.sock`.

To select a non-default Docker server, such as with Docker Swarm, the `withServer()` method should be used.

By passing a URI, and optionally the Credentials ID of a **Docker Server Certificate Authentication** pre-configured in Jenkins, to the method with:

```
node {
    checkout scm

    docker.withServer('tcp://swarm.example.com:2376', 'swarm-certs') {
        docker.image('mysql:5').withRun('-p 3306:3306') {
            /* do things */
        }
    }
}
```

CAUTION

`inside()` and `build()` will not work properly with a Docker Swarm server out of the box

For `inside()` to work, the Docker server and the Jenkins agent must use the same filesystem, so that the workspace can be mounted.

Currently neither the Jenkins plugin nor the Docker CLI will automatically detect the case that the server is running remotely; a typical symptom would be errors from nested `sh` commands such as

```
cannot create /…@tmp/durable-…/pid: Directory nonexistent
```

When Jenkins detects that the agent is itself running inside a Docker container, it will automatically pass the `--volumes-from` argument to the `inside` container, ensuring that it can share a workspace with the agent.

Additionally some versions of Docker Swarm do not support custom Registries.