Buildah Ruby Gem

A comprehensive Ruby wrapper for buildah, providing an object-oriented interface to build OCI container images without requiring Docker or root privileges.

Features

- **Object-oriented API**: Clean, intuitive Ruby interface for buildah operations
- **No Docker required**: Uses buildah's daemonless architecture
- **No root privileges**: Build containers as a regular user
- **Comprehensive coverage**: Supports all major buildah commands
- **Error handling**: Detailed error reporting with custom exception classes
- **Flexible configuration**: Customizable buildah paths and environment variables
- **Well tested**: Comprehensive RSpec and Cucumber test suites

Installation

Add this line to your application's Gemfile:

```
gem 'buildah'
```

And then execute:

```
bundle install
```

Or install it yourself as:

```
gem install buildah
```

Prerequisites

You need to have buildah installed on your system. Please refer to the buildah installation guide for your platform.

To verify buildah is available:

```
buildah version
```

Quick Start

```ruby
require 'buildah'

# Create a new buildah client
client = Buildah.new

# Check if buildah is available
puts "Buildah available: #{Buildah.available?}"
puts "Buildah version: #{Buildah.version}"

# Create a container from an image
container = client.from('alpine:latest')

# Run commands in the container
container.run(['apk', 'add', 'curl'])
container.run('echo "Hello from buildah-ruby!"')

# Configure the container
container.config(
  cmd: 'echo "Hello World"',
  port: '8080',
  workdir: '/app'
)

# Commit the container to create a new image
image = container.commit('my-custom-image:latest')

# Clean up
container.rm
```

Usage Examples

Working with Containers

```ruby
client = Buildah.new

# Create container from scratch
scratch_container = client.from('scratch')

# Create container with specific options
container = client.from('ubuntu:20.04', name: 'my-container', pull: 'always')

# List all containers
containers = client.containers
containers.each do |c|
  puts "Container: #{c.id} (#{c.name}) from #{c.image}"
end

# Run commands with options
container.run(['apt-get', 'update'], user: 'root', workdir: '/tmp')

# Add files to container
container.add('/host/path/file.txt', '/container/path/')
container.copy('/host/path/dir/', '/container/path/', chown: 'user:group')

# Mount and work with filesystem
mount_point = container.mount
# ... work with mounted filesystem
container.umount

# Inspect container
info = container.inspect
puts "Container config: #{info['Config']}"
```

Working with Images

```ruby
client = Buildah.new

# Pull images
image = client.pull('nginx:alpine')

# List images
images = client.images
images.each do |img|
  puts "Image: #{img.name} (#{img.id})"
end

# Tag images
image.tag('my-nginx:v1.0')

# Push images to registry
image.push('registry.example.com/my-nginx:v1.0',
           creds: 'username:password')

# Remove images
image.rmi(force: true)

# Inspect images
details = image.inspect
puts "Image architecture: #{details['Architecture']}"

# Get image history
history = image.history
history.each do |layer|
  puts "Layer: #{layer['Id']} - #{layer['CreatedBy']}"
end
```

Building Images

```ruby
client = Buildah.new

# Build from Dockerfile
image = client.build('.',
                     tag: 'my-app:latest',
                     file: 'Dockerfile.prod',
                     build_arg: ['VERSION=1.0', 'ENV=production'],
                     no_cache: true)

# Build with advanced options
image = client.build('/path/to/context',
                     tag: 'multi-arch:latest',
                     platform: 'linux/amd64,linux/arm64',
                     target: 'production',
                     squash: true,
                     pull: true)

# Use build-using-dockerfile (bud) command
image = Buildah::Builder.bud(client, '.',
                             tag: 'bud-image:latest',
                             layers: true,
                             format: 'oci')
```

Configuration Management

```ruby
client = Buildah.new
container = client.from('alpine')

# Configure container step by step
config = Buildah::Config.new(client)

config.set_env(container.id, { 'APP_ENV' => 'production', 'PORT' => '3000' })
config.set_workdir(container.id, '/app')
config.set_user(container.id, 'appuser:appgroup')
config.set_cmd(container.id, ['./start.sh'])
config.expose_ports(container.id, ['3000', '8080'])

# Add labels and annotations
config.add_labels(container.id, {
  'version' => '1.0.0',
  'maintainer' => 'team@example.com'
})

config.add_annotations(container.id, {
  'org.opencontainers.image.source' => 'https://github.com/example/app'
})
```

Error Handling

```ruby
begin
  client = Buildah.new
  container = client.from('nonexistent:image')
rescue Buildah::BuildahNotFoundError => e
  puts "Buildah not installed: #{e.message}"
rescue Buildah::CommandError => e
  puts "Command failed: #{e.message}"
  puts "Exit code: #{e.exit_code}"
  puts "Stderr: #{e.stderr}"
rescue Buildah::ContainerError => e
  puts "Container operation failed: #{e.message}"
rescue Buildah::ImageError => e
  puts "Image operation failed: #{e.message}"
end
```

Advanced Configuration

```ruby
# Custom buildah path and environment
client = Buildah.new(
  buildah_path: '/usr/local/bin/buildah',
  env: {
    'BUILDAH_ISOLATION' => 'chroot',
    'TMPDIR' => '/custom/tmp'
  },
  debug: true
)

# Get system information
info = client.info
puts "Storage driver: #{info['store']['GraphDriverName']}"
puts "Storage root: #{info['store']['GraphRoot']}"
```

API Reference

Main Classes

- **`Buildah::Client`**: Main interface for buildah operations
- **`Buildah::Container`**: Represents a working container
- **`Buildah::Image`**: Represents a container image
- **`Buildah::Builder`**: Handles building images from Containerfiles
- **`Buildah::Config`**: Manages container and image configuration

Exception Classes

- `Buildah::Error`: Base error class
- `Buildah::BuildahNotFoundError`: Buildah command not found
- `Buildah::CommandError`: Command execution failed
- `Buildah::ContainerError`: Container operation failed
- `Buildah::ImageError`: Image operation failed
- `Buildah::BuildError`: Build operation failed
- `Buildah::ConfigError`: Configuration operation failed

Development

After checking out the repo, run `bin/setup` to install dependencies.

Run the test suite:

```
 # Run RSpec tests
bundle exec rake spec

# Run Cucumber features
bundle exec rake features

# Run all tests and linting
bundle exec rake
```

You can also run `bin/console` for an interactive prompt.

To install this gem onto your local machine, run `bundle exec rake install`.

Contributing

1. Fork the repository
2. Create your feature branch (`git checkout -b my-new-feature`)
3. Make your changes and add tests
4. Ensure all tests pass (`bundle exec rake`)
5. Commit your changes (`git commit -am 'Add some feature'`)

6. Push to the branch (`git push origin my-new-feature`)
7. Create a Pull Request

License

This gem is available as open source under the terms of the Apache License 2.0.

Acknowledgments

- Buildah project for the excellent container building tool
- Podman project for the broader container ecosystem
- All contributors to the Ruby container tooling ecosystem