项目依赖了 prometheus 相关模块：

```
<dependency>
    <groupId>io.prometheus</groupId>
    <artifactId>simpleclient</artifactId>
    <version>${prometheus.version}</version>
</dependency>

<dependency>
    <groupId>io.prometheus</groupId>
    <artifactId>simpleclient_httpserver</artifactId>
    <version>${prometheus.version}</version>
</dependency>

<dependency>
    <groupId>io.prometheus</groupId>
    <artifactId>simpleclient_pushgateway</artifactId>
    <version>${prometheus.version}</version>
</dependency>
```

AbstractPrometheusReporter

实现了 MetricReporter 接口，

PrometheusReporter

继承了 AbstractPrometheusReporter 抽象类，类中有下面两个常量：

```
static final String ARG_PORT = "port";
private static final String DEFAULT_PORT = "9249";
```

表示 port 的常量和默认的 port 为 9249。另外的属性有：

```
private HTTPServer httpServer;
private int port;
```

然后在 open 方法中先从配置中获取到端口，然后根据端口创建 PrometheusReporter 的 HTTP server：

```
public void open(MetricConfig config) {
    super.open(config);

    String portsConfig = config.getString(ARG_PORT, DEFAULT_PORT);
    Iterator<Integer> ports =
NetUtils.getPortRangeFromString(portsConfig);

    while (ports.hasNext()) {
        int port = ports.next();
        try {
            // internally accesses
CollectorRegistry.defaultRegistry
            httpServer = new HTTPServer(port);
            this.port = port;
            log.info("Started PrometheusReporter HTTP server on
port {}.", port);
            break;
        } catch (IOException ioe) { //assume port conflict
            log.debug("Could not start PrometheusReporter HTTP
server on port {}.", port, ioe);
        }
    }
    if (httpServer == null) {
        throw new RuntimeException("Could not start
PrometheusReporter HTTP server on any configured port. Ports: " +
portsConfig);
    }
}
```

HTTPServer 内部的构造函数，从给定的 registry 启动一个 HTTP 服务器，为 Prometheus metrics 提供服务

PrometheusPushGatewayReporter

将 metrics 数据上报给 Prometheus PushGateway 的类，继承自 AbstractPrometheusReporter 抽象类，实现了 Scheduled 接口，有下面三个属性：

```
private PushGateway pushGateway;
private String jobName;
private boolean deleteOnShutdown;
```

open 方法中读取配置和进行初始化：

```
public void open(MetricConfig config) {
    super.open(config);
    //读取配置
    String host = config.getString(HOST.key(),
HOST.defaultValue());
    int port = config.getInteger(PORT.key(), PORT.defaultValue());
    String configuredJobName = config.getString(JOB_NAME.key(),
JOB_NAME.defaultValue());
    boolean randomSuffix =
config.getBoolean(RANDOM_JOB_NAME_SUFFIX.key(),
RANDOM_JOB_NAME_SUFFIX.defaultValue());
    deleteOnShutdown = config.getBoolean(DELETE_ON_SHUTDOWN.key(),
DELETE_ON_SHUTDOWN.defaultValue());

    if (host == null || host.isEmpty() || port < 1) {
        throw new IllegalArgumentException("Invalid host/port
configuration. Host: " + host + " Port: " + port);
    }

    if (randomSuffix) {
        this.jobName = configuredJobName + new AbstractID();
    } else {
        this.jobName = configuredJobName;
    }
    //构建 PushGateway 对象
    pushGateway = new PushGateway(host + ':' + port);
    log.info("Configured PrometheusPushGatewayReporter with {host:
{}, port:{}, jobName: {}, randomJobNameSuffix:{}, deleteOnShutdown:
{}}", host, port, jobName, randomSuffix, deleteOnShutdown);
}
```

在 report 方法中通过 PushGateway 的 push 方法将 默认的注册器和 JobName 作为参数传入：

```
public void report() {
    try {
        pushGateway.push(CollectorRegistry.defaultRegistry,
jobName);
    } catch (Exception e) {
        log.warn("Failed to push metrics to PushGateway with
jobName {}.", jobName, e);
    }
}
```

内部其实是调用了 doRequest 方法：

```
void doRequest(CollectorRegistry registry, String job, Map<String,
String> groupingKey, String method) throws IOException {
String url = gatewayBaseURL + URLEncoder.encode(job, "UTF-8");

if (groupingKey != null) {
  for (Map.Entry<String, String> entry: groupingKey.entrySet()) {
    url += "/" + entry.getKey() + "/" +
URLEncoder.encode(entry.getValue(), "UTF-8");
  }
}
HttpURLConnection connection = (HttpURLConnection) new
URL(url).openConnection();
connection.setRequestProperty("Content-Type",
TextFormat.CONTENT_TYPE_004);
if (!method.equals("DELETE")) {
  connection.setDoOutput(true);
}
connection.setRequestMethod(method);

connection.setConnectTimeout(10 * MILLISECONDS_PER_SECOND);
connection.setReadTimeout(10 * MILLISECONDS_PER_SECOND);
connection.connect();

try {
  if (!method.equals("DELETE")) {
    BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(connection.getOutputStream(), "UTF-8"));
    TextFormat.write004(writer, registry.metricFamilySamples());
    writer.flush();
    writer.close();
  }

  int response = connection.getResponseCode();
  if (response != HttpURLConnection.HTTP_ACCEPTED) {
    throw new IOException("Response code from " + url + " was " +
response);
  }
} finally {
  connection.disconnect();
}
}
```

PrometheusPushGatewayReporterOptions

PrometheusPushGatewayReporter 的配置类，有这些可以提供 host、port、jobName、randomJobNameSuffix、deleteOnShutdown、filterLabelValueCharacters。

注意

如果你是想上报到 Prometheus 或者 Prometheus PushGateway 的话，你需要将 /opt/flink-metrics-prometheus-1.9.0.jar 复制到 /lib 目录下。

然后如果你是上报到 Prometheus 的话，那么你可选的参数有：

```
port: （可选）Prometheus 导出器侦听的端口，默认为 9249。为了能够在一个主机上运行多个报告实例（例如，当一个 TaskManager 与 JobManager 共同使用时），建议使用类似的端口范围 9250-9260。

filterLabelValueCharacters: （可选）：指定是否过滤标签值字符。如果启用，将删除与 [a-zA-Z0-9：_] 不匹配的所有字符，否则不会删除任何字符。 在禁用此选项之前，请确保您的标签值符合 Prometheus 要求
```

示例：

```
metrics.reporter.prom.class:
org.apache.flink.metrics.prometheus.PrometheusReporter
```

Flink metrics 类型映射到 Prometheus metrics 类型，如下所示：

Flink	Prometheus	Note
Counter	Gauge	Prometheus counters cannot be decremented.
Gauge	Gauge	Only numbers and booleans are supported.
Histogram	Summary	Quantiles .5, .75, .95, .98, .99 and .999
Meter	Gauge	The gauge exports the meter's rate.

如果你想上报到 Prometheus PushGateway 的话，那么你可以选择配置的参数有下面这些：

键	默认	描述
deleteOnShutdown	true	指定是否在关闭时从PushGateway中删除指标。
Host	(none)	PushGateway服务器主机。
jobName	(none)	将推送指标的作业名称
port	-1	PushGateway服务器端口。
randomJobNameSuffix	true	指定是否应将随机后缀附加到作业名称。

配置示例:

```
metrics.reporter.promgateway.class:
org.apache.flink.metrics.prometheus.PrometheusPushGatewayReporter
metrics.reporter.promgateway.host: localhost
metrics.reporter.promgateway.port: 9091
metrics.reporter.promgateway.jobName: zhisheng
metrics.reporter.promgateway.randomJobNameSuffix: true
metrics.reporter.promgateway.deleteOnShutdown: false
```