```
import os
from typing import Any, Dict


import click
import yaml
from loguru import logger
import pulumi
from pulumi_gcp import cloudrun


# Configure logging
logger.add("deploy_agent.log", rotation="10 MB", retention="10 days", level="INFO")



# Function to load and validate agent.yaml
def load_yaml(file_path: str) -> Dict[str, Any]:
    try:
        logger.info(f"Loading YAML configuration from {file_path}")
        with open(file_path, "r") as file:
            config = yaml.safe_load(file)
            logger.info("YAML configuration loaded successfully")
            return config
    except FileNotFoundError as e:
        logger.error(f"YAML file not found: {e}")
        raise
    except yaml.YAMLError as e:
        logger.error(f"Error parsing YAML: {e}")
        raise



# Function to check file existence

def check_file_exists(file_path: str) -> bool:

    if os.path.exists(file_path):

        return True

    else:

        logger.error(f"File not found: {file_path}")

        return False



# Function to build and push Docker image

def build_and_push_docker_image(

    agent_name: str, project_id: str, dockerfile_path: str

) -> str:

    image_name = f"gcr.io/{project_id}/{agent_name}"

    try:

        logger.info(f"Building Docker image: {image_name}")

        build_command = f"docker build -t {image_name} -f {dockerfile_path} ."

        if os.system(build_command) != 0:

            raise RuntimeError(f"Failed to build Docker image: {image_name}")


        logger.info(f"Pushing Docker image: {image_name}")

        push_command = f"docker push {image_name}"

        if os.system(push_command) != 0:
            raise RuntimeError(f"Failed to push Docker image: {image_name}")

        logger.info(f"Docker image built and pushed successfully: {image_name}")
        return image_name
    except Exception as e:
        logger.error(f"Error building or pushing Docker image: {e}")
        raise


@click.command()
@click.option("--project-id", envvar="GCP_PROJECT_ID", help="Google Cloud Project ID")
@click.option(
    "--region",
    default="us-central1",
    envvar="GCP_REGION",
    help="Region where the agent will be deployed",
)
@click.option(
    "--dockerfile-path", default="./Dockerfile", help="Path to the Dockerfile"
)
@click.option("--yaml-path", default="./agent.yaml", help="Path to the agent YAML file")
def deploy_agent(dockerfile_path: str, yaml_path: str) -> None:
    try:
        project_id = os.getenv("GCP_PROJECT_ID")
        region = os.getenv("GCP_REGION")
```

```
# Step 1: Validate existence of Dockerfile and agent.yaml
if not check_file_exists(dockerfile_path):
    logger.error("Dockerfile is missing, exiting.")
    return

if not check_file_exists(yaml_path):
    logger.error("YAML file is missing, exiting.")
    return

# Step 2: Load the YAML configuration
config = load_yaml(yaml_path)
agent_name = config.get("agent_name", "default-agent")
config.get("description", "No description provided")
resources = config.get("cloud_run", {}).get("resources", {})
environment_variables = config.get("cloud_run", {}).get(
    "environment_variables", {}
)

# Step 3: Initialize Pulumi programmatically
logger.info(f"Setting up Pulumi with project {project_id} and region {region}")
pulumi.runtime.set_config("gcp:project", project_id)
pulumi.runtime.set_config("gcp:region", region)

# Step 4: Build and push Docker image
docker_image = build_and_push_docker_image(
    agent_name, project_id, dockerfile_path
```

```
)

# Step 5: Deploy to Google Cloud Run
logger.info(f"Deploying {agent_name} to Google Cloud Run...")
cloud_run_service = cloudrun.Service(
    agent_name,
    location=region,
    template={
        "spec": {
            "containers": [
                {
                    "image": docker_image,
                    "resources": {
                        "limits": {
                            "cpu": resources.get("cpu", "1"),
                            "memory": resources.get("memory", "512Mi"),
                        }
                    },
                    "env": [
                        {"name": k, "value": v}
                        for k, v in environment_variables.items()
                    ],
                }
            ]
        }
    },
            autogenerate_name=True,
        )

        logger.info(
            f"Agent {agent_name} deployed successfully at: {cloud_run_service.status.url}"
        )
        print(f"Agent deployed at: {cloud_run_service.status.url}")

    except Exception as e:
        logger.error(f"An error occurred during deployment: {e}")
        raise


# if __name__ == "__main__":
#     deploy_agent()
```