```
SafeguardsCodec(
    codec=codec,  # any numcodecs codec
    safeguards=[
        dict(
            kind="qoi_eb_stencil",
            qoi="""
                V["earth_radius"] = 6371000;  # [m], globally averaged

                # extract u and v from the stacked (first) dimension
                V["u"] = X[0, I[1], I[2]];
                V["v"] = X[1, I[1], I[2]];
                # convert latitude in degrees to radians
                V["latRad"] = c["lat"] * pi / 180;

                # (1) along the longitude axis, we handle the 360 degree periodic
                #     boundary
                V["dVdPhi"] = finite_difference(
                    V["v"],
                    order=1, accuracy=2, type=0, axis=2,
                    grid_centre=c["lon"], grid_period=360,
                );

                # approximate the spatial derivatives with finite differences
                # (2) along the latitude axis, we need to handle the polar boundary
                V["dUdTheta"] = where(
                    abs(c["lat"]) < 90,
                    # use central difference where possible
                    finite_difference(
                        V["u"] * cos(V["latRad"]),
                        order=1, accuracy=2, type=0, axis=1,
                        grid_centre=c["lat"],
                    ),
                    # at the poles, use forward/backwards difference
                    where(
                        c["lat"] == -90,
                        finite_difference(
                            V["u"] * cos(V["latRad"]),
                            order=1, accuracy=1, type=-1, axis=1,
                            grid_centre=c["lat"],
                        ),
                        finite_difference(
                            V["u"] * cos(V["latRad"]),
                            order=1, accuracy=1, type=+1, axis=1,
                            grid_centre=c["lat"],
                        ),
                    ),
                );

                # compute the relative vorticity
                return (V["dVdPhi"] - V["dUdTheta"]) / (
                    V["earth_radius"] * cos(V["latRad"])
                );
            """,
            type="abs",
            eb=1e-8,
            neighbourhood=[
                # [u, v]: stacked variables
                dict(axis=0, before=0, after=1, boundary="valid"),
                # latitude: edge boundary is good enough since we ensure in the
                #           QoI that the edge-extended values are not used
                dict(axis=1, before=1, after=1, boundary="edge"),
                # longitude: wrapping boundary
                dict(axis=2, before=1, after=1, boundary="wrap"),
            ],
        )
    ],
    # provide the latitude and longitude dimensions as late-bound fixed constants
    # and ensure that they can be broadcast to the data shape
    fixed_constants=dict(
        lat=latitude.reshape(1, -1, 1), lon=longitude.reshape(1, 1, -1),
    ),
)
```